Proyecto Fin de Grado

DISEÑO Y DESARROLLO DE UNA APLICACIÓN CLIENTE-SERVIDOR PARA LA GESTIÓN DE GRUPOS DE TRABAJO EN UN ENTORNO EDUCATIVO COLABORATIVO ORIENTADO A DISPOSITIVOS IOS

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES

Tutor: ESTÉVEZ AYRES, IRIA MANUELA
Alumno: POZO SANTIAGO, JUAN MANUEL

Índice

Listas

1. Lista de ilustraciones

2. Lista de tablas

Acrónimos

APK	Application PacKage File, paquete específico para el sistema operativo Android.
Apple Inc.	Apple Incorporation.
DTD	Document Type Definition
CGI	Common Gateway Interface
GPL	GNU General Public License
ID de Apple	Identificación especifica y personal como usuario de Apple.
iOS	iPhone Operating System.
IPA	iPhone Application, formato de archivos utilizado en aplicaciones Apple.
JSON	JavaScript Object Notation
Mac OS X	Mac Operating System X.
MIME	"*Multipurpose Internet Mail Extensions*", en español "extensiones multipropósito de correo de internet".
NIA	Número de Identificación de Alumno
SGML	"Standard Generalized Markup Language" o "Estándar de Lenguaje de Marcado Generalizado".
SQL	Structured Query Language.
URL	Uniform Resource Locator
XML	Extensible Markup Language

Capítulo 1. Introducción

1.1. Situación actual

Con el sistema de enseñanza universitaria actual, el 'Plan Bolonia', prima el trabajo del alumno y en gran medida los trabajos en grupo para fomentar el trabajo colectivo.

Los alumnos se enfrentan a trabajos en grupo en la mayoría de asignaturas. En cada una de ellas, los grupos estarán integrados por diferentes personas que, en muchos casos, no serán de la elección de los propios estudiantes, sino que serán impuestos por el cuerpo docente. La relación entre los integrantes no es un factor en la elección, provocando en muchos casos que los grupos estén formados por personas que nunca han trabajado juntas o que incluso nunca haya existido trato entre ellas.

La organización es una de las claves en el éxito de los trabajos en grupo, más si cabe cuando los integrantes de dicho grupo no están acostumbrados a trabajar entre ellos.

El primer paso para un grupo de trabajo es ponerse en contacto, algo que en ocasiones no es trivial. El desconocimiento del resto de integrantes del grupo puede suponer un desafío para esta primera toma de contacto. No siempre se dispone de los nombres de los alumnos, normalmente en los listados de grupos sólo se proporcionan los identificadores de alumno genéricos.

En un ambiente de trabajo, generalmente los trabajadores no quieren facilitar sus datos personales (correo electrónico, número de teléfono, etc.), más incluso cuando se ven forzados a trabajar con gente desconocida. Es preferible proporcionar datos únicamente académicos.

Para facilitar esta tarea y la posterior organización general en los grupos, se propone una solución que ayude a los alumnos a ponerse en contacto y a establecer una comunicación continuada entre los integrantes del grupo.

1.2. Estructura de la memoria

Inicialmente, se ofrece un índice general de la memoria y un listado de todas las tablas e ilustraciones incluidas. Además, se incluye un listado de acrónimos utilizados durante la memoria.

En el capítulo de "Estado del arte" se presentan las diferentes partes del proyecto y de las opciones existentes para desarrollar cada una de ellas.

En "Diseño" se exponen los requisitos del Proyecto y la estructura de cada una de las partes definidas en el capítulo de "Estado del arte".

Una vez establecido el diseño general de la aplicación, se procede a detallar la manera en que se desarrollo en el capítulo de "Implementación".

Con todos los contenidos implementados, se procede a una fase de comprobaciones recogidas en el capítulo "Pruebas".

Todas las partes que incluyen el desarrollo del proyecto ya se han mostrado en los capítulos interiores. En el resto de la memoria se detalla la planificación seguida y el presupuesto (capítulo de "Presupuesto"), conclusiones obtenidas durante el periodo de desarrollo del Proyecto (capítulo de "Conclusiones") y, por último, un listado detallado de los recursos consultados durante todo el periodo del Proyecto (capitulo de "Bibliografía").

Capítulo 2. Estado del arte

2.1. Introducción

Para la elaboración del proyecto, se exponen las diferentes partes que será necesario implementar.

Para cada una de las partes que se deben implementar, existen diferentes opciones. En este capítulo se exponen las principales, destacando las ventajas que ofrecen al Proyecto. En la Ilustración 1 se pueden apreciar las partes que deberán ser implementadas y que serán discutidas en este capítulo.

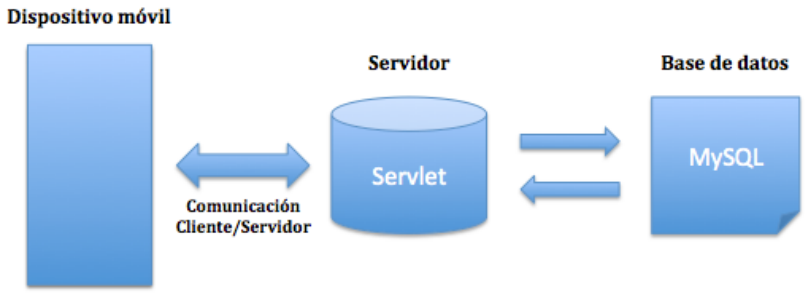

Ilustración 1. Partes que se implementarán

2.2. Sistema operativo del cliente

Existen multitud de sistemas operativos entre los que podemos elegir: iOS, Android, Symbian (producto de la unión de diferentes empresas de telefonía móvil, entre las que podríamos destacar: Samsung, Nokia, Siemens o Motorola), BlackBerry OS… Aunque cuando hablamos de telefonía móvil no podemos evitar pensar en los dos 'gigantes' del mercado: **iOS** y **Android**.

Android es actualmente el sistema operativo con mayor cuota de mercado[2]. Además, todas sus aplicaciones están basadas en lenguaje Java y cualquiera desde cualquier ordenador puede desarrollar una aplicación. Todo programador puede probar fácilmente sus propias aplicaciones en los dispositivos generando un archivo ".apk"(*Application PacKage File*, paquete específico para el sistema operativo Android) e instalándolo en el móvil.

iOS en cambio, no ofrece las mismas facilidades para el desarrollador. En primer lugar, un futuro desarrollador de iOS deberá disponer de un equipo con una versión mínima de Mac OS X 10.6.6 (la versión 10.6 de Mac OS X es conocida como 'Snow Leopard'). Esta versión es la primera que permite la descarga del programa 'Xcode', el entorno de desarrollo integrado de 'Apple Inc.' que permite programar en iOS. Aunque existen otros programas diferentes que ofrecen la posibilidad de desarrollar en iOS, si queremos poder probar nuestras aplicaciones necesitaremos el 'Xcode' para generar los archivos '.ipa' (iPhone Application, formato de archivos utilizado en aplicaciones Apple) que permiten la instalación de la aplicación en el teléfono móvil.

Para toda descarga de aplicaciones(incluida la descarga de 'Xcode') o incluso para poder iniciar cualquier dispositivo móvil de 'Apple Inc.' (como son: iPhone, iPod Touch e iPad) necesitaremos una identificación específica de Apple ('ID de Apple') que nos permitirá, además, el acceso a contenidos de Apple en Internet. Esta identificación es gratuita, aunque se puede utilizar para descargar contenido de pago a través de ella.

El ID de Apple únicamente nos permite acceder a contenidos; para poder desarrollar los mismos, Apple nos obliga además a estar registrado en su web como desarrollador, más concretamente en su web de 'Apple Developer'. Sin estar registrado como desarrollador, podremos programar nuestras aplicaciones en el 'Xcode', aunque nunca podremos probarlas en dispositivos reales. Nuestras pruebas se verán limitadas al simulador que incluye el propio entorno de desarrollo. El registro en 'Apple Developer' como tal, no nos permitirá probar nuestras aplicaciones en dispositivos iOS, deberemos obtener un certificado de desarrollador una vez registrados, descargarnos ese certificado en nuestro 'Xcode' y generar las aplicaciones con dicho certificado. Este certificado ya no es gratuito y deberá abonarse una cantidad anualmente para mantener el certificado activo y con ello mantener nuestras aplicaciones útiles; ya que, una vez caducado nuestro certificado, las aplicaciones generadas con él dejarán de funcionar.

Si el desarrollador quiere las aplicaciones para sí mismo, no necesita nada más, pero si lo que quiere es que todo usuario de Apple pueda disfrutar de sus aplicaciones necesita pasar otro control por parte de Apple. A diferencia de Android, donde se pueden subir las aplicaciones al 'Market' (servicio disponible en todos los dispositivos Android que permite la búsqueda y descarga de otras aplicaciones) con libertad, Apple posee un último escalón antes de poder publicar tus aplicaciones en el 'App Store' (como su homólogo en Android, es el servicio de iOS que permite la búsqueda y descarga de otras aplicaciones). Para poder publicar una aplicación, ésta debe pasar un proceso de validación por parte de Apple antes de ser publicada.

Como se puede ver, Apple dispone de un control mucho más férreo sobre el uso que los usuarios pueden ejercer de su entorno. Aunque pueda parecer excesivo, también se nos asegura con ello que lo que podemos encontrar y descargar en el 'App Store' ha pasado un proceso de validación, evitándonos además un exceso de

aplicaciones, cosa que no ocurre en Android. Desde el punto de vista del autor del proyecto, una de las razones del éxito de los dispositivos Apple radica en este control, la seguridad que sienten los usuarios.

2.3. Comunicación Cliente/Servidor

Los clientes deben interactuar con un servidor. Gracias al servidor, los clientes pueden comunicarse entre ellos y compartir información.

2.3.1. JSON

JSON[4](JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Es un lenguaje simple de escribir y de simple procesamiento por parte de la máquina, lo que favorece su uso en aplicaciones con gran volumen de datos.

Está constituido por dos estructuras[3]:

- Se envían objetos con una clave cada uno. Cada valor tendrá su clave. En el envío se establece una clave para cada objeto, y en recepción se obtiene el objeto a través la clave.

- Se pueden enviar listas de valores, todos ellos asignados a una misma clave. Estas listas suelen ser *'arrays'* (una colección de valores) o vectores.

Todos los lenguajes de programación son capaces de reconocer esta estructura.

Los JSON se pueden representar como[3]:

- Objetos: valores asignados a una única clave. Todo el objeto JSON se enmarca entre llaves ("{" y "}"). Para enviar más de un objeto, se separan por comas. Cada objeto tiene la siguiente estructura:

  ```
  { "clave1": "valor1" , "clave2": "valor2" }
  ```

- Extendiendo el caso anterior, se puede enviar un grupo de valores incluidos en un *'array'*, todos ellos asignados a una misma clave. Para indicar los valores incluidos en el *array* se comienza abriendo un corchete, y se termina cerrando con otro.

  { "claveArray" : ["valor1", "valor2"] }

- Se puede enviar directamente un *String* en el objeto. Un *String* o cadena de caracteres es una secuencia de longitud variable de números, letras o signos o símbolos.

2.3.2. XML

XML[5] (Extensible Markup Language) es un metalenguaje que fue diseñado básicamente para estructurar, almacenar e intercambiar datos entre diferentes aplicaciones. Fue desarrollado por el *World Wide Web Consortium* (W3G)[6].

Características principales[5]:

- Simple: simplifica la escritura con respecto a HTML o SGML ("Standard Generalized Markup Language" o "Estándar de Lenguaje de Marcado Generalizado", es un sistema de organización y etiquetado de documentos).

- Extensible: cualquiera puede inventar sus propias etiquetas para marcar cualquier tipo de documento. Las etiquetas, además, pueden ser compartidas.

- Estándar abierto: deriva del lenguaje SGML. Muchas herramientas permiten recrearlo, manejarlo o implantarlo en un ordenador y distribuirlo.

- Eficiente: utiliza algoritmos para reutilizar fragmentos de documentos. De esta manera, sólo tiene que enviarlos una vez.

- Basado en la experiencia: ha sido diseñado por personas que tienen amplia experiencia en este tipo de lenguajes, aplicándose dicha experiencia al desarrollo del mismo.

- Consensuado: el diseño incluye las opiniones de organismos coordinadores de HTML y SGML, así como de personas que han desarrollado importantes aplicaciones con estos estándares.

- Libre: nadie tiene la propiedad o patente de XML, por lo que su uso no implica pagos.

- Internacional: es compatible con prácticamente todos los alfabetos.

- Manejable: incluye métodos para declarar y reforzar estructuras documentales usadas actualmente, como las bases de datos.

XML tiene dos estructuras[7]:

- Física: se compone por unidades llamadas *entidades*. Una *entidad* puede contener otras *entidades*. Todo documento XML comienza con una *entidad* documento conocida como "raíz".

- Lógica: se compone de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos ellos identificados.

Además, un documento XML debe estar "bien formado"; lo que quiere decir que, debe cumplir todas las especificaciones respecto a las reglas sintácticas y tener la estructura jerárquica bien definida. También, debe ser "válido", siguiendo una estuctura y semántica adecuada definida en un *DTD* (Document Type Definition).

2.4. Servidor

Se necesita un proveedor de información para el cliente. De acuerdo a la función que vaya a realizar, se pueden establecer dos tipos de servidor[8]:

- Servidor 'dedicado': dedica toda su potencia a atender las solicitudes del cliente.

- Servidor no 'dedicado': no se centra en las solicitudes del cliente, sino que se dedica también a las acciones que se puedan desarrollar localmente.

El servidor utilizado será 'dedicado', únicamente atenderá las necesidad de nuestro cliente.

2.4.1. Java Servlet

Un Servlet[9][10] es un objeto java que extiende a javax.servlet.http.HttpServlet. Existen muchos tipos de Servlet, pero el HttpServlet con diferencia es el más utilizado. Corren dentro y fuera de un contenedor de servlets (un ejemplo de ellos son los *Tomcat*) y además extienden su funcionalidad. Un contenedor servlet procesa las peticiones del cliente y las redirecciona a un objeto Servlet.

Funcionamiento de un contenedor de Servlets[10]:

- El cliente realiza una solicitud al servidor HTTP que es un contenedor de Servlets.

- El contenedor de Servlets pasa la petición a un Servlet para que la controle.

- El Servlet se encarga de generar la respuesta que se entrega al contenedor de Servlets.

- El contenedor devuelve la respuesta al cliente.

2.4.2. Interfaz de entrada común

Interfaz de entrada común[11] (en inglés "Common Gateway Interface", abreviado CGI) es una tecnología de la World Wide Web que permite a un cliente solicitar datos de un programa ejecutado en un servidor web. Fueron una de las primeras en crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web; de tal manera que, esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.

A continuación, se describe la forma de actuación de un CGI de forma esquemática[11]:

- En primer lugar, el cliente envía una petición al servidor donde incluye una URL ("Uniform Resource Locator") y se comprueba si se trata de una invocación de un CGI.

- Después, el servidor prepara el entorno para ejecutar la aplicación donde la información procede principalmente del cliente.

- El servidor ejecuta la aplicación, capturando su salida estándar.

- A continuación, la aplicación realiza su función: como consecuencia de su actividad se genera un objeto MIME (*Multipurpose Internet Mail Extensions*) que la aplicación escribe en su salida típica.

- Para acabar, el servidor envía la información producida al terminar la aplicación, unida a información del propio servidor, al cliente. Durante el proceso, el cliente se mantiene a la espera de la respuesta. La aplicación debe especificar el tipo de objeto MIME que se genera (campo CONTENT_TYPE).

2.5. Base de datos

Una base de datos es un conjunto de datos almacenados de manera lógica para su posterior uso[12].

Las principales características de las bases de datos son[13]:

- Independencia lógica y física de los datos.

- Redundancia mínima.

- Acceso por parte de múltiples usuarios.
- Integridad de los datos
- Consultas complejas optimizadas.
- Seguridad de acceso.
- Acceso a través de lenguajes de programación estándar.

2.5.1. MySQL

MySQL[16] es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

La licencia para el uso de MySql es gratuita, ya que la empresa propietaria de la mayor parte del copyright patrocina su uso. Para empresas que quieran incorporarlo en productos privativos, deben pagar una licencia especial para su utilización.

MySQL es una base de datos muy rápida en la consulta de datos, aunque no tanto cuando se modifican dichos datos.

MySQL[14] distribuye los datos en tablas en vez de archivarlos todos en un conjunto global. Esto favorece la rapidez de la búsqueda y la flexibilidad de la base de datos.

Características principales[16]:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas(relaciones entre columnas de otras tablas).
- Conectividad segura.
- Replicación. Se generan copias en varios discos, lo que beneficia la velocidad en la lectura de los datos. Además, cuando los datos están

siendo modificados, si se produce un error, podemos disponer de una de las otras copias.

- Búsqueda e indexación de campos de texto. Los datos son fácilmente localizables.

2.5.2. MongoDB

MongoDB[15] es un sistema de base de datos NoSQL (difieren del modelo clásico del sistema de gestión de bases de datos relacionales en aspectos importantes, el más destacado es que no usan SQL como el principal lenguaje de consultas) orientado a documentos.

MongoDB es de código abierto, se distribuye y desarrolla libremente.

MongoDB forma parte de la familia de sistemas de base de datos NoSQL[18], que no guardan la información en tablas relacionadas entre sí, sino que utiliza estructuras de datos en documentos de tipo JSON, lo cual beneficia la integración en determinadas aplicaciones.

Las características principales de MongoDB son[15]:

- Consultas *Ad hoc*[19]: permite diseñar las consultas al usuario, no está limitado a las consultas preestablecidas.

- Indexación: todos los campos pueden ser indexados y permiten la creación de interconexiones entre los datos.

- Replicación: al igual que MySql, permite las copias de la información en nodos "esclavos", que permiten únicamente la lectura de datos. Si se necesita la modificación de los mismos, se precisa el acceso al nodo principal. Por ello, favorece la lectura frente a la escritura.

- Balanceo de carga: se puede ejecutar en múltiples servidores para poder distribuir la carga de trabajo y permitir que el sistema siga funcionando en caso de un fallo de software.

- Almacenamiento de archivos: gracias a la ventaja que ofrece el balanceo de carga y la replicación de datos, puede ser utilizado como un sistema de almacenamiento de archivos. Además, incluido en el software de MongoDB, se incluyen funciones para la manipulación de los archivos.

- Ejecución de JavaScript del lado del servidor: permite incluso el envío de código JavaScript a la base de datos, desde la que se ejecutará directamente.

2.6. Conclusiones

En función de las características del proyecto, se deciden los componentes que lo formarán:

- Cliente: el sistema operativo del cliente será iOS por exigencias del proyecto.

- Comunicación cliente/servidor[4]: por simplicidad de código para la interpretación de la comunicación se utilizará JSON. Para interpretar la comunicación XML es necesaria mayor complejidad de código y de esfuerzo computacional. Dadas la simplicidad de las consultas que se requerirán, JSON es una opción más adecuada.

- Servidor[20]: los CGI arrancan un proceso para cada petición HTTP; mientras que, los Servlets mantienen el proceso arrancado. De esta manera, para peticiones simples y rápidas suponen un coste mayor en CGI que en los Servlets, ya que el proceso de arrancada puede ser mayor que el de ejecución. Por ello, los Servlets se ajustan más a las condiciones del proyecto.

  Además, como se precisa un contenedor de servlets, se desplegará un Apache Tomcat donde se instalará el Servlet.

- Base de datos[15,16,18]: MongoDB ofrece la posibilidad de una base de datos escalable y fácil de migrar en caso de que fuera necesario. MySql, por otra parte, ofrece las ventajas de ser una base de datos basada en SQL. La simplicidad de utilizar SQL y la posibilidad de estructurar los datos en tablas que se relacionan provocan que MySql sea la opción elegida.

Capítulo 3. Diseño

3.1. Introducción

Se propone la implementación de una aplicación que permita la gestión de grupos de prácticas de una manera fácil y sencilla. La aplicación proporcionará un servicio de mensajería para cada grupo y un sistema de citas. Sólo los alumnos pertenecientes a un grupo podrán acceder a sus contenidos de chat y citas.

Se deberá implementar una aplicación móvil, un servidor, una base de datos y las comunicaciones entre ellos.

3.2. Objetivos iniciales

Para la realización de este Proyecto se establecen unos requisitos mínimos que se deben cumplir.

3.2.1. Requisitos mínimos generales

El planteamiento inicial del Proyecto incluía los siguientes requisitos generales:

1. El proyecto debe abarcar la implementación de la aplicación, el servidor y la base de datos.

2. La implementación del servidor y la base de datos debe ser compatible con móviles de distintos sistemas operativos.

3.2.2. Requisitos mínimos de la aplicación

La aplicación deberá incluir en su implementación unos contenidos y utilidades mínimas:

1. Login inicial de usuario.
    a. Es necesario un usuario incluido en la base de datos para poder acceder a las ventajas que ofrece la aplicación.
    b. Los campos no pueden estar vacíos al realizar la petición.
    c. Si el alumno es válido se avanza al listado de grupos, si no lo es, se informa de que es inválido.
2. Listado de grupos.
    a. Los grupos a los que se pertenece se muestran en la pantalla principal. Si no existen grupos, se muestra vacío.
    b. Desde esta pantalla se debe acceder a la creación de nuevos grupos.
    c. Desde esta pantalla se debe acceder a la pantalla de peticiones pendientes.
    d. Si existen peticiones de grupo pendientes, debe indicarse visualmente en el botón desde el que se accede a las peticiones pendientes.
3. Creación de grupo.
    a. El alumno creador debe estar incluido siempre en el grupo.
    b. Deben introducirse sólo alumnos válidos a un grupo.
    c. El campo de alumno no puede estar vacío al intentar introducirlo.
    d. Cuando se añade un alumno, debe mostrarse en la lista de añadidos.
    e. No se puede introducir dos veces el mismo alumno.
    f. Comprobar que el nombre del grupo se ha introducido al crearlo.
4. Peticiones pendientes de grupo.
    a. Debe mostrarse como un listado de la misma manera que los grupos.
    b. Al seleccionar uno, debe mostrar un mensaje de aceptación. En caso afirmativo, el grupo que fue seleccionado se eliminará de la tabla de 'Pendientes' y se añadirá a la tabla de 'Grupos'. En caso negativo, no se altera nada.
5. Grupo.
    a. Un grupo debe tener un 'Chat'.
    b. Un grupo debe tener un gestor de Citas (una 'Agenda').
    c. Un grupo debe tener una sección de 'Información'.
    d. Desde cualquiera de las tres secciones se debe poder volver al listado de grupos inicial.
6. Chat.
    a. Todo el trafico de información debe realizarse en función del grupo en el que se encuentre.
    b. Cuando se accede a la zona de 'Chat' se deben solicitar todos los mensajes pendientes a la base de datos. Una vez devueltos los mensajes, se deben eliminar de la base de datos.
    c. El listado de mensajes se debe mostrar encabezado por el nombre del usuario que envía cada mensaje.
    d. Se solicitarán los mensajes pendientes periódicamente. El tiempo de refresco debe ser corto para simular la instantaneidad.
    e. Cuando se envíe un mensaje, debe introducirse en la tabla de mensajes pendientes un registro por cada alumno perteneciente al grupo.

   f. Todos los mensajes ya recibidos se almacenarán localmente para evitar sobrecarga al servidor.
7. Agenda.
   a. Todo el trafico de información debe realizarse en función del grupo en el que se encuentre.
   b. Las citas ya creadas se mostrarán en un listado.
   c. Cuando se seleccione una cita, se mostrará toda la información de la misma.
   d. Para crear una cita nueva todos los campos deben estar rellenados.
   e. No se puede crear una cita con un nombre repetido.
   f. Al crear una nueva cita, ésta será incluida en el listado.
8. Información de grupo.
   a. Se mostrará el nombre del grupo y el nombre de todos los integrantes.
   b. Se debe poder añadir nuevos alumnos al grupo. Al añadirse, se incluirán en la lista de peticiones pendientes.
   c. Debe existir la posibilidad de abandonar el grupo. Al abandonarlo se debe eliminar el archivo donde se almacenan los mensajes del 'Chat'. Además, todos los registros de la base de datos referentes al alumno y el grupo deben eliminarse.
   d. Si al abandonar el grupo, éste queda vacío, en la base de datos se eliminarán todos los registros de mensajes pendientes, peticiones pendientes de grupo y citas.

Los objetivos especificados exigen la implementación de diferentes partes muy diferenciadas. La Ilustración 2 muestra el esquema general de las partes que componen nuestro objetivo.

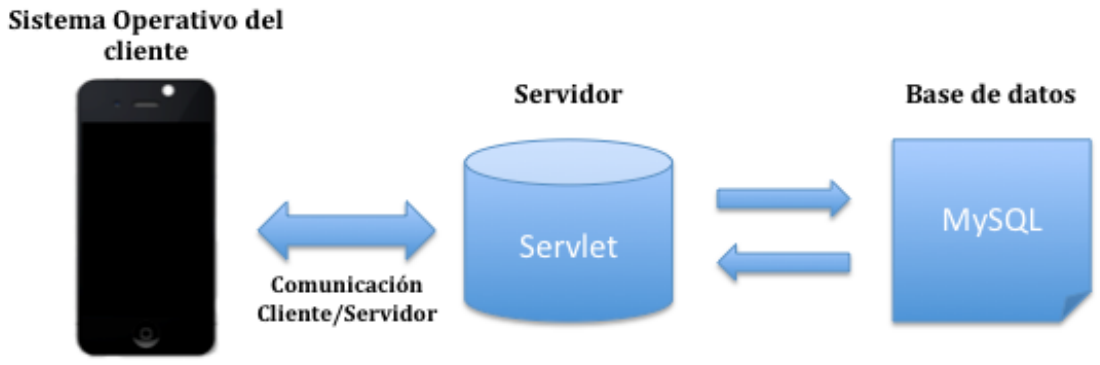

Ilustración 2. Esquema general de funcionamiento

Para todas ellas, se debe elegir entre diferentes opciones para su desarrollo. Dicha elección se detalla en el punto 2.6. de este Proyecto.

3.3. Diseño general de la aplicación

Con el objetivo de resultar sencillo y de fácil utilización, se establece un diseño simple estructurado en dos partes:

- Proceso de '*Login*' y gestión de los grupos.

  Inicialmente, se solicita al alumno que introduzca un Número de Identificador de Alumno o 'NIA', junto con la contraseña. Será necesario un alumno válido para poder acceder a los contenidos de la aplicación.

  Según muestra la Ilustración 3, a partir del proceso de 'Login', la aplicación se centra en una pantalla que contiene un listado de los grupos activos. A partir de esta pantalla de grupos, se pueden añadir nuevos grupos o aceptar grupos que otros alumnos hayan añadido.

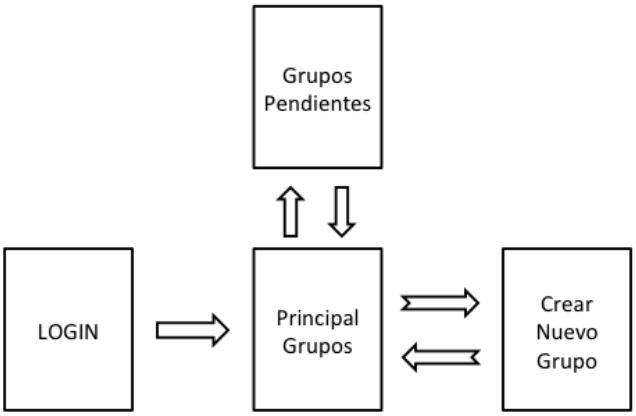

Ilustración 3. Diseño general de la aplicación

Para crear nuevos grupos y para aceptar grupos creados por otros alumnos se accede a otras pantallas. Cuando se crea un nuevo grupo, sólo se creará el grupo para el alumno que lo crea, para el resto de alumnos se creará una petición en la base de datos; de manera que un alumno sólo pertenecerá a un grupo que haya creado o que haya aceptado previamente. En la Ilustración 4, se muestra las acciones que se llevan a cabo en cada pantalla.

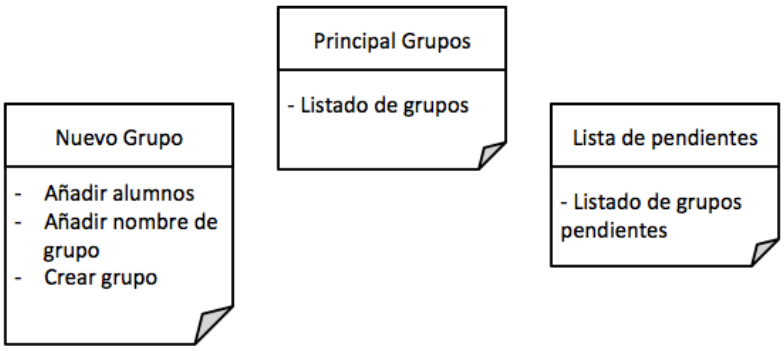

Ilustración 4. Acciones generales en las pantallas

En la pantalla que muestra la lista de peticiones de grupos pendientes, únicamente tendremos el listado de peticiones. Sólo se podrán aceptar grupos o volver al listado de grupos principal.

Para la creación de nuevos grupos será necesario introducir los alumnos y un nombre de grupo. Se podrá tener dos grupos con el mismo nombre; por lo tanto, los grupos no deben regirse por los nombres, sino por un identificador interno que el usuario no vea. La razón principal para dar la posibilidad de múltiples grupos con el mismo nombre, radica en que un alumno no tiene que tener conocimiento de los nombres de los grupos de sus compañeros, lo que provocaría muchos problemas a la hora de que los alumnos añadidos aceptasen la petición.

- Contenido de un grupo.

Desde la pantalla Principal de grupos se accede al contenido específico de cada grupo. Este contenido incluye un Chat, un gestor de citas y una pantalla desde la que se pueda ver la información y gestión general del grupo.

Inicialmente se accede a la pantalla de Chat y desde ella se puede acceder a la agenda o a la información general del grupo. Las diferentes opciones dentro de un grupo (chat, agenda e información) son accesibles desde las otras opciones. Además, desde todas las pantallas se accede al listado de grupos principal.

En la Ilustración 5, se ve el flujo general de la aplicación, cómo el contenido propio de un grupo se relaciona entre sí. Desde cualquiera de ellos se puede volver al listado de grupos.

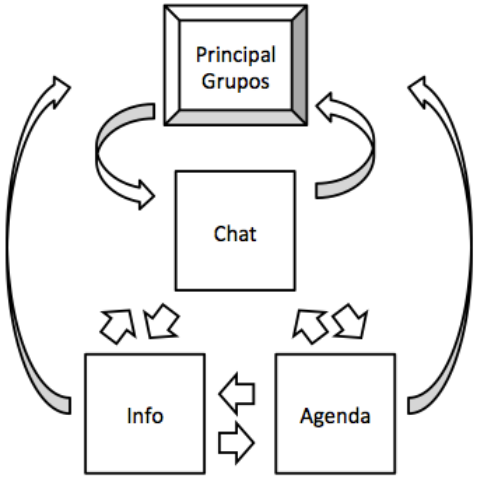

Ilustración 5. Diseño de los flujos

En la Ilustración 6, se muestran los contenidos de las diferentes secciones. El chat muestra mensajes enviados entre los componentes del grupo. La agenda permite almacenar citas para organizar el grupo. La

información muestra el nombre del grupo, los alumnos pertenecientes al mismo, la posibilidad de añadir compañeros y la posibilidad de abandonarlo.

Ilustración 6. Contenido de las secciones de un grupo

3.4. Diseño del servidor

El servidor gestiona las comunicaciones entre el dispositivo móvil y la base de datos.

En la Ilustración 7, se muestran los diferentes tipos de gestión que lleva a cabo el servidor.

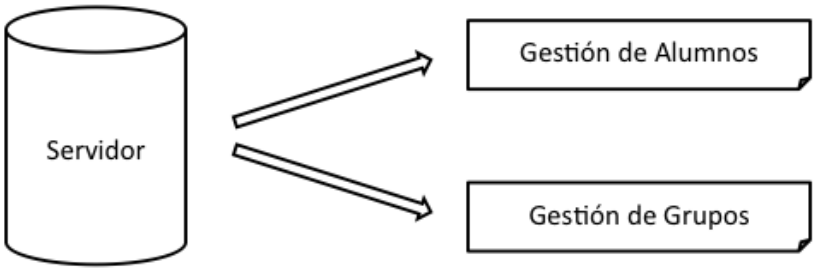

Ilustración 7. Gestiones del servidor.

- La "Gestión de Alumnos" se encarga de las acciones relativas a los alumnos, como el '*login*'.

- La "Gestión de grupos" se encarga de las acciones relativas a los grupos, como la creación de nuevos grupos, creación de citas, etc.

Para cada acción de la aplicación que precise de la colaboración del servidor, se enviarán peticiones para que se realicen dichas acciones. Para ello, es necesario un establecimiento de conexión entre el dispositivo y el servidor. La Ilustración 8 muestra un esquema general del proceso de conexión entre un dispositivo y el servidor.

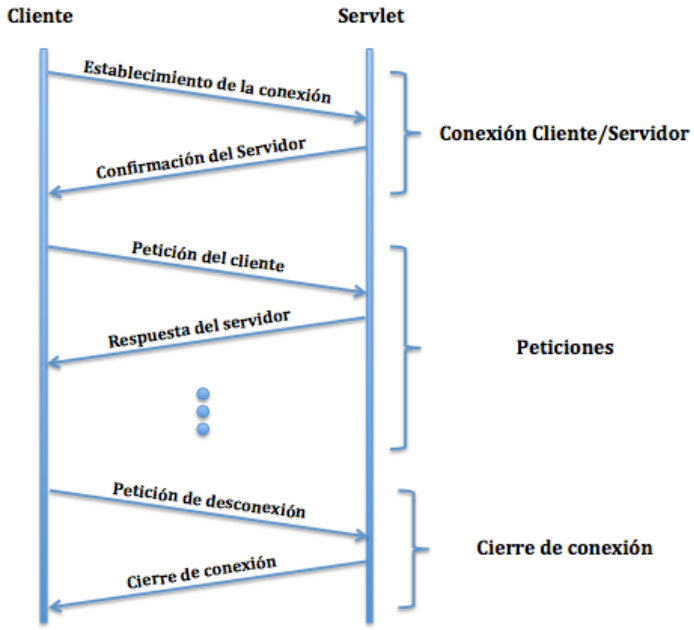

Ilustración 8. Esquema de conexión Cliente/Servidor

3.5. Diseño de la base de datos

La base de datos almacena toda la información de los grupos y de los alumnos.

Dentro de la base de datos, existen diferentes tablas que almacenan la información. Existe una tabla especifica para cada contenido (Ilustración 9):

- Alumnos: información referente a los alumnos.
- Grupos: información de los grupos ya creados.
- Pendientes: grupos creados pero que no han sido confirmados por los alumnos.
- Chat: mensajes que no se han recibido aún. Los ya recibidos se almacenan localmente en cada dispositivo.
- Agenda: almacenaje de las citas de cada grupo.

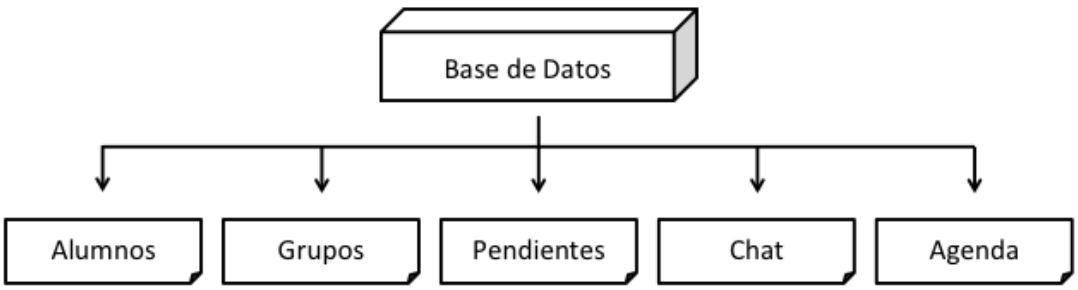

Ilustración 9. Estructura de la base de datos

El servidor realizará consultas en cada acción que realice la aplicación. La aplicación realiza una petición al servidor y el servidor consulta la base de datos. Ésta devuelve los datos solicitados y el servidor los envía a la aplicación.

Las consultas a bases de datos siguen el esquema mostrado en el diseño del servidor como se muestra en la Ilustración 10.

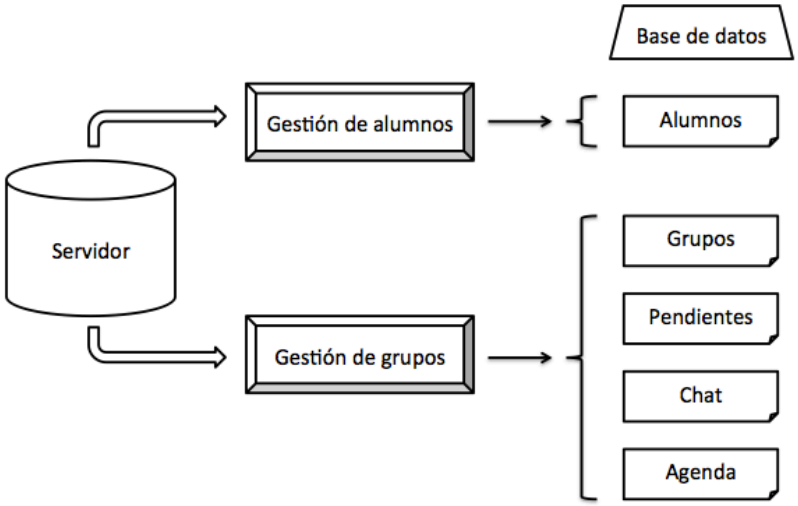

Ilustración 10. Consultas a la base de datos según el esquema del servidor

Capítulo 4. Implementación

Con el diseño definido, se procede al desarrollo de las diferentes partes del proyecto. Inicialmente, se detallan los equipos (incluidas sus características) de los que se dispone para el desarrollo de todas las partes. Seguidamente, se abordarán individualmente todas las partes de la implementación.

4.1. Requisitos de Pruebas

Para el desarrollo del proyecto se dispuso de las siguientes herramientas:

- Programación iOS:
  - Equipos:
    - MacBook Pro principios del 2011.
      - Versión: Mac OS X Lion 10.7.4.
      - Procesador: 2 GHz Intel Core i7.
      - Memoria: 8GB 1333 MHz DDR3.
    - MacMini mediados del 2011.
      - Versión: Mac OS X Lion 10.7.5.
      - Procesador: 2.7 GHz Intel Core i7.
      - Memoria: 8 GB 1333 MHz DDR3.
  - Programa utilizado para desarrollar el código: Xcode.
    - Versión inicial: 4.3.3.
    - Versión final: 4.6.1.
  - Simulador iOS (incluido en Xcode):
    - Versión: 6.0.
- Servidor:
  - Equipo:
    - MacBook Pro principios del 2011 (partición de Windows).
      - Partición de disco duro: Windows 7
  - Máquina Virtual (Oracle VM VirtualBox).
    - Versión de VirtualBox: 4.2.8.
    - Oracle Java 7 (JDK).
    - Sistema Operativo: Ubuntu 12.04.
    - Memoria base: 1024MB.
    - Memoria de vídeo: 64MB.
    - Tamaño Virtual: 10GB.
  - Apache tomcat 7.0.37
  - Eclipse IDE for Java EE Developers.

4.2. Fases de Implementación

Para la implementación general, el desarrollo se realizó en tres partes:

- **Implementación de la interfaz simple**.

  En primer lugar, la aplicación no dependía de ningún servidor. Todos los datos necesarios para reproducir los comportamientos habituales que tendría la aplicación estaban disponibles localmente.

  La aplicación consta de dos partes claramente diferenciadas (explicadas en los puntos 4.2.1. y 4.2.2.). Cada una de estas partes se desarrollan inicialmente por separado para discriminar los problemas generados por cada sección. Una vez implementadas correctamente, se procede a su unión.

- **Implementación del servidor y su relación con la aplicación**.

  Con la aplicación funcionando de manera local, se procede a la elaboración del servidor y a las comunicaciones con la aplicación.

  Dicho servidor se implementará como un Java Servlet, con objetos Json como comunicación con el cliente.

  El servidor se configura, en un primer lugar, para recibir peticiones sencillas sin tener en cuenta la base de datos. Básicamente, se comprueba que existe comunicación entre la aplicación y el servidor. Para ello, no se utiliza la aplicación que funciona de manera local, sino que se desarrolla una aplicación simple aparte que realice peticiones al servidor.

  Una vez comprobada la comunicación entre el servidor y la aplicación, se procede a la inclusión del código en la aplicación que funcionaba localmente y a realizar las consultas a la base de datos desde el servidor.

  Como base de datos utilizamos MySql por sencillez de instalación en el sistema.

- **Arreglos y mejoras**.

  Con todo incluido, se realizan comprobaciones generales de funcionamiento. En ellas, se descubren errores de funcionamiento y comportamientos no contemplados durante el diseño. Los errores se solucionan y se introducen mejoras de funcionamiento.

4.3. Estructuración de la aplicación

Por el funcionamiento de la aplicación, existen dos partes claramente diferenciadas desde el punto de vista de la estructura:

- Contenidos generales.

  Esta parte de la aplicación se dedica a la visualización general de grupos. Desde ella podremos ver todos los grupos a los que pertenecemos, crear nuevos o aceptar grupos en los que hemos sido añadidos. Se accederá a través de un 'login' al iniciar la aplicación. En la Ilustración 11, se puede ver el esquema que tiene.

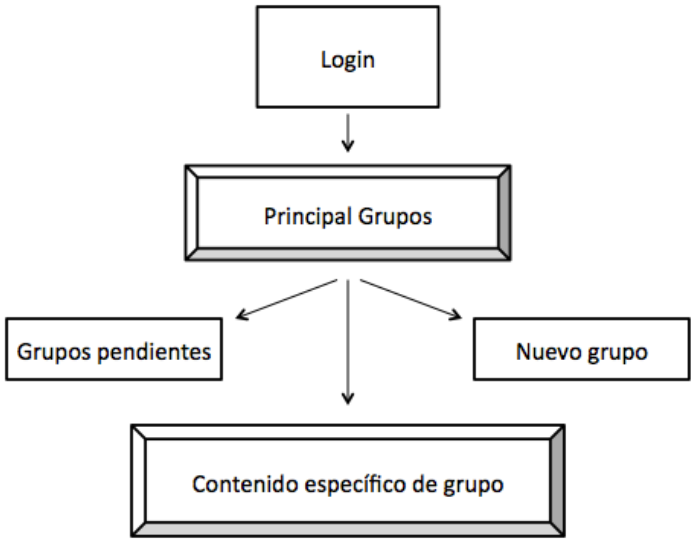

Ilustración 11. Esquema de los contenidos generales

- Contenidos específicos.

  La información concreta de cada grupo incluye un chat, un gestor de citas e información general del grupo (donde se incluyen los nombres de los integrantes y desde donde se pueden añadir nuevos miembros o abandonar el grupo).

  Desde cada sección se puede volver a acceder al listado de grupos o a cualquiera de las otras secciones específicas de grupo.

  En la Ilustración 12, se ve cómo desde el listado de grupos se accede al contenido específico y cómo desde el contenido específico se puede volver al listado.

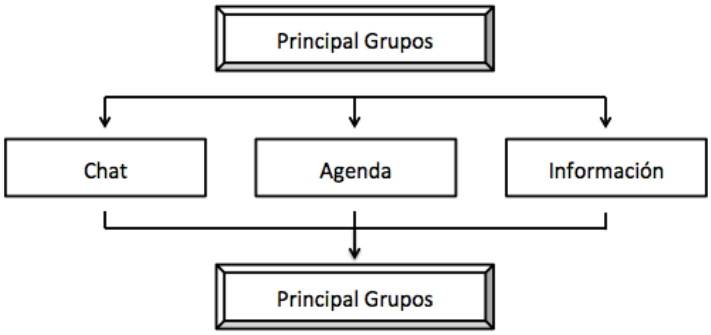

Ilustración 12. Flujo listado de grupo - contenido específico

En la Ilustración 13 se ve un ejemplo del flujo desde una pantalla de contenido específico, en este caso desde la pantalla de Agenda. Como se puede ver, se puede acceder al resto de contenidos o al listado principal.

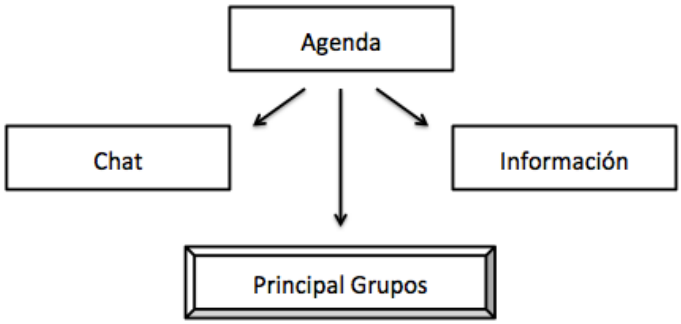

Ilustración 13. Ejemplo del flujo dentro del contenido específico de grupo

4.3.1. Contenidos generales

4.3.1.1. Pantalla de Registro

La aplicación posee un proceso inicial de '*login*', quedando restringido el acceso únicamente para los alumnos de la Universidad Carlos III. En caso de no pertenecer a la Universidad, se nos rechazará desde la propia pantalla de inicio.

Desde la pantalla de '*login*', podremos también acceder a una pantalla de información donde encontramos los datos tanto del desarrollador como del tutor del proyecto.

Desde el momento en que pasamos el proceso de registro, toda la aplicación tiene en cuenta el usuario, cargando contenidos específicos del mismo. Estos contenidos están almacenados en la base de datos del servidor.

Una vez realizado el proceso de '*login*', la primera pantalla que veremos será un listado de los grupos activos que tenemos.

4.3.1.2. Pantalla de Grupos

La visión principal consta de un listado de los grupos asignados a nuestro usuario, desde el que podremos acceder al contenido específico de cada uno de ellos.

Además del listado de grupos, desde aquí se accede a las pantallas de creación de nuevos grupos y a la de solicitudes de grupo pendientes.

- **Nuevos grupos**: Desde esta opción, se ofrece la posibilidad de crear un nuevo grupo.

  En primer lugar, se deben añadir los alumnos de uno en uno. Una vez añadidos todos, con el campo de 'Nombre de grupo' rellenado, se crea el nuevo grupo. Al crear el grupo, la aplicación nos devuelve a la pantalla de grupos principal.

  Existe cierto nivel de control a la hora de generar un nuevo grupo. Las restricciones son las siguientes:

  - Campos vacíos: no se permitirá dejar vacío el campo de 'Nuevo Alumno' a la hora de añadir uno. Tampoco se podrá dejar vacío el campo de 'Nombre de grupo' cuando aceptemos las modificaciones. En los dos casos, se mostrará un mensaje de alerta indicando dónde se encuentra el error.

  - Alumnos duplicados: para evitar errores de procesado en la base de datos, cada vez que queramos añadir un alumno al listado de integrantes del nuevo grupo, se realiza una comprobación entre los alumnos ya incluidos (el creador del grupo se incluye inicialmente) para que no se añada uno que ya esté. De la misma manera que con los campos vacíos, la aplicación avisará por medio de una alerta del problema encontrado.

  - Alumno no existe: el servidor debe comprobar que el alumno introducido existe en la base de datos. Si no existe, se motrará un mensaje de alerta.

- **Peticiones pendientes**: cuando otros usuarios crean un nuevo grupo, éste no aparece directamente en los listados de grupos de los alumnos. Cada alumno por separado deberá aceptar estar incluido en dicho grupo antes de poder acceder a sus contenidos específicos.
  Cuando tengamos una petición de grupo, en la pantalla del listado de grupos aparecerá un icono indicándolo.
  Las peticiones se ordenan en un listado que nos muestra el nombre del grupo. Seleccionando uno, se nos da la opción de pertenecer definitivamente al grupo, momento en el cual será eliminado del listado de

'Grupos Pendientes' y añadido al listado de 'Grupos' en la pantalla principal.

Cuando se quiera acceder al contenido de uno de los grupos, no hay más que seleccionarlo y se accede.

Una característica importante de los grupos es que no se guían por el nombre, sino que cada grupo lleva un identificador asociado que es el que la aplicación realmente controla. Todos los contenidos asociados a un grupo se asocian al identificador en vez de al nombre. De esta manera, se evitan errores por grupos con el mismo nombre o similares. El identificador de grupo es único y lo asigna el servidor cuando incluye las referencias del mismo en la base de datos.

Es importante señalar que los usuarios que tengan un grupo en la sección de 'Pendientes', no podrán tener acceso a sus contenidos específicos hasta que no lo haya aceptado.

4.3.2. Contenidos específicos de un grupo

Cuando se selecciona un grupo accedemos a una sección totalmente distinta de la aplicación. Cada grupo posee tres secciones, pudiendo acceder a cada una de ellas desde cualquiera de las otras dos. Estas secciones son:

- Chat.
- Agenda.
- Información.

4.3.2.1. Chat

Es la primera sección que se ve cuando se selecciona un grupo. Desde esta, sección los usuarios pueden comunicarse a través de un servicio de mensajería instantánea.

El funcionamiento general de la base de datos ha intentado evitar en lo posible el almacenamiento de todos los comentarios en la base de datos. Por ello, una vez que el usuario ha recibido los comentarios los guardará en un fichero local del dispositivo, donde irá almacenando los mensajes que haya recibido hasta el momento. De esta manera, no se hace necesario el almacenamiento de grandes cantidades de información en la base de datos y es fácilmente controlable por parte del cliente.

Cuando un alumno comente en el Chat, su comentario se añadirá a la base de datos, con una referencia para cada usuario al que vaya destinado (los integrantes del grupo, incluido el propio alumno que comenta).

Cuando un usuario acceda a la sección de Chat, el cliente solicita al servidor los mensajes de ese grupo. El servidor consultará la base de datos y devolverá todos los comentarios que el usuario no haya recibido hasta ahora.

El usuario recibirá todos los mensajes pendientes, los moldeará para incluirlos en el fichero local y los guardará. Inmediatamente después de guardar los comentarios en el fichero, leerá éste cargando todos los mensajes y mostrándolos por pantalla empezando por el más reciente.

Cada vez que el usuario envíe un comentario, solicitará los mensajes pendientes, donde estará incluido el suyo propio, y procederá a incluirlos en el fichero. Incluyendo referencias a uno mismo en la base de datos, evitamos la necesitad de incluir datos en el fichero de dos maneras diferentes: a través de la base de datos o a través del botón 'Enviar', siempre incluiremos los mensajes a través de la base de datos.

Como no siempre se envían mensajes, o se entra y sale del Chat, la sección posee una función que solicita los nuevos mensajes al servidor y los muestra periódicamente.

4.3.2.2. Agenda

La organización es la base de todo trabajo en grupo. Desde esta sección es desde donde podremos controlar las reuniones del equipo.

La sección consta de dos zonas principales:

- **Citas**: en la parte superior de la pantalla se ofrece un listado con las citas programadas. Cuando seleccionamos una, se nos muestra una alerta con todo el contenido de la alerta:
  - Asunto.
  - Lugar.
  - Fecha.

- **Nuevas citas**: en la mitad inferior de la pantalla, disponemos de los campos que debemos rellenar para generar la nueva cita. Los campos son:
  - Asunto: Razón de la cita. Además, servirá de título para la cita en el listado superior de las ya creadas.
  - Lugar: Localización de la reunión.
  - Fecha: Día de la reunión. Para rellenar este campo, cuando pulsamos sobre el campo se nos proporciona un seleccionable por día, mes y año que facilita la elección de la fecha. Además, también evita incorrecciones y unifica el formato.

  Además, los campos de las nuevas citas incluyen la restricción de los espacios en blanco, evitando por tanto que una cita quede sin detallar. Tampoco podrán existir dos citas con el mismo "Asunto".

Para las citas antiguas, el servidor dispone de una comprobación periódica que eliminará las citas que superen la fecha especificada en las mismas. Todos los días

el servidor realiza una comprobación de las fechas de las citas, si alguna posee una fecha pasada, ésta se borrará de la base de datos y ya no será visible para el usuario.

Todas las citas llevan asignado el identificador de grupo al que pertenecen. El identificador se envía junto con los detalles de la cita desde el cliente al servidor y éste la crea. Al entrar en la sección, los usuarios hacen la petición de las citas con el identificador de grupo.

4.3.2.3. Información

La última de las pantallas propias de un grupo muestra principalmente a los integrantes.

En primer lugar, se muestra el nombre completo del grupo, no siendo modificable. Inmediatamente después, se ofrece un listado de los alumnos pertenecientes al grupo.

Podremos añadir alguno más en caso de olvido en el momento de la creación o si el alumno se unió tardíamente al grupo de prácticas. El proceso de adhesión de un nuevo alumno es similar al descrito a la hora de crear un nuevo grupo. Se deberá añadir al alumno en el campo correspondiente y, al aceptar a dicho alumno, se realizarán las convenientes comprobaciones:

- Campos vacíos.
- Alumnos duplicados.
- Alumno no existe.

Una vez realizadas las comprobaciones, se añade al alumno a la lista de 'Peticiones Pendientes'.

La última de las opciones que ofrece esta pantalla es la posibilidad de abandonar el grupo. Para ello, seleccionaremos el botón "Abandonar grupo", y con ello eliminaremos el grupo de nuestra lista, además de eliminar también el fichero donde almacenamos el Chat del grupo.

Si cuando eliminamos el grupo somos el último de los integrantes, informamos además al servidor de que debe eliminar todo rastro de citas, peticiones de grupo, mensajes de Chat pendientes y registros asociados a dicho grupo, quedando completamente borrado de la base de datos.

4.4. Peticiones al servidor

Las peticiones a un Servlet comúnmente incluye dos únicos tipos que son:
- Peticiones 'GET'.
- Peticiones 'POST'.

Mientras que las peticiones 'GET' son peticiones genéricas enviadas al servidor, las peticiones 'POST' se usan para enviar datos al servidor y que éste los procese.

Como ya se ha indicado, la aplicación consta de un proceso inicial de '*login*', lo cual condiciona todas las peticiones siguientes. Como el servidor es un elemento común, no podemos tratarlo de manera específica. Por lo tanto, en cada petición que realicemos deberemos indicar quién la realiza. En consecuencia, el servidor que utiliza nuestra aplicación constará principalmente de peticiones de tipo 'POST'; ya que, siempre habrá que enviar, como mínimo, el usuario que realiza la petición.

Existe una única petición de tipo 'GET' configurada. Es una petición de control que sirve para asegurarse de que el servidor está activo. Esta petición nunca la solicitará el cliente, se podrá realizar desde cualquier navegador y únicamente mostrará un mensaje de confirmación para hacer saber al solicitante que el Servlet se encuentra activo y escuchando.

Para realizar las peticiones de datos al servidor, el cliente debe ejecutar dos tareas (ejemplos mostrados: proceso de *'login'*):

- En primer lugar, se almacena la información que se desea enviar como petición en un paquete. Una vez realizado el paquete de información, se intenta convertir en un paquete de tipo *'Json'* para enviarlo al servidor. Una vez convertido, se envía al servidor llamando a la segunda tarea. Este proceso se puede ver en la Ilustración 14.

```
-(id)setJsonFromData:(NSString *)alumno{

    //Disctionary objetc
    NSArray *keys = [NSArray arrayWithObjects:@"username",
        @"password", nil];
    NSArray *objects = [NSArray arrayWithObjects:self.nombre.text,
        self.password.text, nil];
    NSMutableDictionary *jsonDictionary = [NSDictionary
        dictionaryWithObjects:objects forKeys:keys];

    //is convertable to json object??
    if ([NSJSONSerialization isValidJSONObject:jsonDictionary])
    {
        NSError *error = nil;
        NSData * result = [NSJSONSerialization dataWithJSONObject:
            jsonDictionary options:NSJSONWritingPrettyPrinted error:
            &error];

        if (error == nil && result != nil) {
            NSString *comprobacion =[self jsonPostRequest:result];
        }
        return nil;
    }
}
```

Ilustración 14. Formación del paquete Json en el cliente

- Una vez creado el *'Json',* enviamos la petición al servidor. Para ello, especificamos la URL a la que pertenece. Añadimos a dicha dirección la petición específica (en el caso del ejemplo, es "login"). Para enviar una petición, entre la URL y la petición se añade el símbolo '?' que indica al servidor que lo que le sigue es el nombre propio de la petición. De esta manera, el servidor es capaz de discernir entre unas peticiones u otras.

  En el mismo método, el cliente debe esperar la respuesta del servidor y almacenarla.

  El proceso completo se puede ver en la Ilustración 15.

```
- (id)jsonPostRequest:(NSData *)jsonResquestData{

    NSString *direccion =[NSString stringWithFormat:@"%@?login",
        url_base];

    //url for the request
    NSURL *url = [NSURL URLWithString:direccion];

    //The request
    NSMutableURLRequest *request = [NSMutableURLRequest
        requestWithURL:url cachePolicy:
        NSURLRequestUseProtocolCachePolicy timeoutInterval:64];

    //bind request with jsonRequest
    [request setHTTPMethod:@"POST"];
    [request setValue:@"application/json"
        forHTTPHeaderField:@"Accept"];
    [request setValue:@"application/json"
        forHTTPHeaderField:@"Content-Type"];
    [request setValue:[NSString stringWithFormat:@"%d",
        [jsonResquestData length]] forHTTPHeaderField:@"Content-
        length"];
    [request setHTTPBody:jsonResquestData];

    //send sync request
    NSURLResponse *response = nil;
    NSError *error = nil;
    NSData *result = [NSURLConnection sendSynchronousRequest:request
        returningResponse:&response error:&error];

    if (error == nil){
        //Buscamos la respuesta del Servlet
        NSError *jsonParsingError = nil;
        NSMutableArray *jsonArray = [NSJSONSerialization
            JSONObjectWithData:result options:
            NSJSONReadingMutableContainers|
            NSJSONReadingAllowFragments error:&jsonParsingError];

    }
    return nil;
}
```

Ilustración 15. Envío de la petición y recepción de la respuesta en el cliente.

4.5. Base de datos

La estructura de la base de datos busca simplificar al máximo las búsquedas y facilitar la compresión de los campos. Se utilizará MySql ya que no se precisa una base de datos dinámica; es decir, los registros tienen un tamaño definido y nunca se modifica. Además, las consultas son muy sencillas y fáciles de realizar desde el servidor.

La base de datos se estructura de manera que existen tablas a las que sólo se accederá desde determinadas secciones.

4.5.1. Gestión de usuarios

El control de los usuarios se gestiona desde una única tabla de la base de datos, la tabla 'USERS' cuya estructura se puede ver en la Tabla 1.

Tabla 1. Bases de Datos. Tabla 'USERS'

Columna	Id	Name	Pass
Tipo	INTEGER	CHAR	CHAR

La tabla 'USERS' tiene la característica principal de que en ningún momento se puede modificar desde la aplicación, es una tabla únicamente de consulta. Es la única que tiene esta característica.

El objetivo de la tabla es almacenar los datos personales del alumno, tanto el nombre (que será el identificador de alumno de la Universidad) como su contraseña. El identificador que se le añade no tiene relevancia con respecto a la aplicación, ya que nunca se consulta.

Esta tabla se utiliza sobre todo para el proceso de registro al arrancar la aplicación, donde se comprueba que el nombre introducido corresponda con la contraseña dada. Además, se utiliza cuando se intentan añadir los alumnos a los grupos realizando una simple comprobación de que el alumno introducido pertenece a la tabla.

4.5.2. Gestión de grupos

La gestión de los grupos precisa de dos tablas en la base de datos, una que controle los grupos creados y otra que gestione las solicitudes de grupo.

Los grupos ya creados se almacenan en la tabla 'GROUPS' de la base de datos. Su estructura se muestra en la Tabla 2.

Tabla 2. Bases de Datos. Tabla 'GROUPS'

Columna	Id	Grupo	User
Tipo	INTEGER	CHAR	CHAR

La tabla 'GROUPS' almacena una relación de grupos con alumnos. Existirá un registro para cada alumno que pertenezca a un grupo; es decir, si en un grupo existen varios alumnos, la tabla tendrá una entrada para cada alumno. De esta manera, cuando un alumno abandone un grupo bastará con eliminar el registro propio del alumno con el grupo.

Además, se introduce un identificador que será único para cada grupo. Este identificador lo asigna el propio servidor, buscando el último asignado e incrementándolo en una unidad. Es de gran importancia, ya que todos los procesos en los grupos se basan en este identificador y no en el nombre. Posibilita, entre otras cosas, la existencia de dos grupos con el mismo nombre para un mismo alumno sin que ello provoque errores en las llamadas.

Esta tabla es utilizada para generar los listados de grupos, para consultar nombres de grupos y cuando se elimina un alumno de un grupo.

El control de las solicitudes pendientes en la gestión de los grupos se lleva a cabo desde la tabla 'PENDIENTES' de la base de datos. Su estructura se detalla en la Tabla 3.

Tabla 3. Bases de Datos. Tabla 'PENDIENTES'

Columna	Id_grupo	Grupo	User
Tipo	INTEGER	CHAR	CHAR

Un alumno no tiene necesidad de aceptar pertenecer a un grupo al que es añadido. Para controlar los grupos a los que se añade un alumno, se crea la tabla 'PENDIENTES'. En ella se almacenarán los alumnos que han sido invitados a un grupo, y el nombre al cuál han sido invitados.

Contiene el identificador del grupo al que le han invitado, así como el nombre del mismo y el alumno invitado.

Disponiendo del identificador de grupo no parece necesario disponer también del nombre, pero a la hora de solicitar el listado de peticiones pendientes de un alumno o cuando añadimos una petición a la tabla de 'GROUPS' sí es necesario. De no incluirlo, sería necesario una consulta extra a la base de datos para obtener dicho nombre para:

- Mostrar el listado de peticiones: el identificador nunca se muestra en la aplicación, lo almacenamos, pero mostramos su nombre asociado.
- Añadir a la tabla 'GROUPS': con el identificador no basta, necesitamos el nombre del grupo para completar la tabla.

Esta tabla se utiliza en la creación de grupos, y en la generación del listado de peticiones. Además, cuando un alumno abandona un grupo siendo el último, las peticiones para ese grupo vacío también se eliminan.

4.5.3. Gestión de la Agenda de un grupo

Sólo una tabla es necesaria para controlar los eventos que se muestran en la agenda de un grupo. La tabla sigue la estructura mostrada en la Tabla 4.

Tabla 4. Bases de Datos. Tabla 'AGENDA'

Columna	Id_grupo	Lugar	Asunto	Fecha
Tipo	INTEGER	CHAR	CHAR	CHAR

Cada una de las citas que se creen en la aplicación se almacenará aquí.

La estructura de la tabla trata de ser todo lo simple posible; por ello, no almacena más que los campos que se van a mostrar. Únicamente es necesario el identificador de grupo para clasificarlas. Eso sí, existe una restricción que se aplica en el cliente. Esta restricción evita que se creen dos citas con el mismo identificador y el mismo "Asunto". A diferencia de los grupos, para las citas no existe un identificador único para cada una de ellas, por lo que se guían por el "Asunto". Si existieran dos con el mismo asunto en el mismo grupo, cuando el cliente consultara la cita, habría una de ellas a la que nunca accedería, ya que siempre se devolvería la primera.

4.5.4. Gestión del chat

Los mensajes se almacenan en cada dispositivo, pero para la comunicación entre usuarios es necesario un paso por la base de datos. Para ello, se crea la tabla 'CHAT', que muestra en la Tabla 5.

Tabla 5. Bases de Datos. Tabla 'CHAT'

Columna	Id_grupo	User	Destino	Fecha	Mensaje
Tipo	INTEGER	CHAR	CHAR	CHAR	CHAR

Todos los mensajes que se envíen en el 'Chat' deberán pasar por esta tabla. Si un mensaje se encuentra en esta tabla, querrá decir que el usuario marcado como "Destino" no lo habrá recibido todavía. En el momento en que un usuario consulte la tabla para recibir un chat, éste será borrado.

4.6. Comunicación del Cliente con el servidor

El cliente envía la petición de conexión al servidor y una vez establecida comienza a solicitar los valores que necesita. Habitualmente, cuando solicita una información, inmediatamente después cierra la conexión con el servidor a la espera de que el usuario haga una petición por sí mismo y abra una nueva sesión (pulsando algún botón, seleccionando una celda de una lista, etc.). En general, la comunicación del cliente con el servidor se limita a una única petición.

Para cada comunicación con el servidor, el cliente debe enviar peticiones específicas en función de los contenidos que precise. Para ello, se definen los diferentes 'comandos' que puede recibir el Servlet. En ellos, se define en cada campo el parámetro que enviamos en el paquete Json. Son valores que deben establecerse previamente, ya que el nombre que se dé en un extremo de la comunicación debe darse exactamente igual en el otro extremo, sino no serán capaces de entenderse cliente y servidor.

4.6.1. Comando para logarse

El comando CMD_LOGIN se envía durante el proceso de registro para certificar que el alumno introduce un nombre y una contraseña válidas. En la Tabla 6 se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 6. Comando CMD_LOGIN Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno
password	CHAR	Contraseña del alumno

En la Tabla 7, se pueden ver los parámetros enviados por el servidor después de consultar la base de datos.

Tabla 7. Comando CMD_LOGIN Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	CHAR	"OK" si el *login* es correcto. "NOK" si algo ha fallado o el usuario no existe o la contraseña es incorrecta.

4.6.2. Comando de obtención de grupos

El comando CMD_GROUPS se envía en la carga de la lista de grupos, para recibir el listado de los grupos de la base de datos a los que pertenece el usuario. En la Tabla 8, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 8. Comando CMD_GET_GROUPS Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno

En este caso, el servidor no devuelve una respuesta única, sino que envía una serie de respuestas, todas ellas unidas en un 'JsonArray' (sucesión de objetos 'Json' unidos en un 'Array'). El formato de estas respuestas se puede ver en la Tabla 9. Cada una de ellas corresponderá con la información de cada grupo al que pertenece el usuario dado por el cliente.

Tabla 9. Comando CMD_GET_GROUPS Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo
groupName	CHAR	Nombre del grupo

4.6.3. Comando para saber si existen peticiones

El comando CMD_CHECK_PETITIONS se envía cuando se accede a la página principal del listado de grupos. Se utiliza para saber si existen peticiones pendientes de grupo para el alumno que acaba de registrarse. Se envía el alumno en cuestión para la consulta de la base de datos. En la Tabla 10, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 10. Comando CMD_CHECK_PETITIONS Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno

El servidor devuelve si existen o no peticiones, aunque no especifica cuáles, simplemente informa de su existencia. El modelo del envío se puede ver en la Tabla 11.

Tabla 11. Comando CMD_ CHECK_PETITIONS Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si existen peticiones. "NOK" en caso contrario.

4.6.4. Comando que devuelve peticiones de grupo

El comando CMD_PETITIONS se envía para recibir el listado de peticiones del usuario cuando se accede a la pantalla de 'Peticiones Pendientes'.

En la Tabla 12, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 12. Comando CMD_GET_PETITIONS Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno

En este caso, el servidor no devuelve una respuesta única, sino que envía una serie de respuestas, todas ellas unidas en un 'JsonArray' (sucesión de objetos 'Json' unidos en un 'Array'). El formato de estas respuestas se puede ver en la Tabla 13. Cada una de ellas corresponderá con la información de cada grupo del que el usuario ha recibido petición.

Tabla 13. Comando CMD_GET_ PETITIONS Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo
groupName	CHAR	Nombre del grupo

4.6.5. Comando de aceptación de petición

El comando CMD_ACCEPT_PETITIONS se envía cuando al seleccionar una petición de la lista de 'Peticiones Pendientes' se acepta; de manera que pase a formar parte de los grupos del alumno.

El servidor recibe el nombre del alumno, el identificador de grupo y el nombre del grupo y lo inserta en la base de datos 'GRUPOS'. Además, elimina también la petición que acepta. En la Tabla 14, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 14. Comando CMD_ACCEPT_PETITIONS Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno
groupID	INTEGER	El identificador único de grupo
groupName	CHAR	Nombre del grupo

Si todo el proceso se realiza con éxito, el servidor devuelve una confirmación. La respuesta se puede ver en la Tabla 15.

Tabla 15. Comando CMD_ACCEPT_ PETITIONS Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si el proceso se realiza con éxito. "NOK" en caso contrario.

4.6.6. Comando de comprobación de alumno

El comando CMD_CHECK_PARTNER es necesario para saber si un alumno existe en la base de datos. Antes de ser añadido a la lista de alumnos que pertenecerán al grupo, se comprueba que dicho alumno realmente existe. Para ello, únicamente es necesario enviar el alumno al servidor para comprobar que efectivamente existe, como se puede ver en la Tabla 16.

Tabla 16. Comando CMD_CHECK_PARTNER Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno

El servidor realizará una simple comprobación y, en caso de existir, enviará el parámetro mostrado en la Tabla 17.

Tabla 17. Comando CMD_ CHECK_PARTNER Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si el alumno existe. "NOK" en caso contrario.

4.6.7. Comando que añade un grupo

El comando CMD_INSERT_GROUP se envía cuando se crea un nuevo grupo para que éste sea insertado en la base de datos. En la Tabla 18, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 18. Comando CMD_INSERT_GROUP Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno
groupName	CHAR	Nombre del grupo

El servidor devuelve el identificador de grupo del grupo que acaba de crear. Es necesario enviar este dato; ya que, para la posterior creación de las peticiones de grupo para el resto de alumnos, es indispensable. El modelo del envío se puede ver en la Tabla 19.

Tabla 19. Comando CMD_INSERT_GROUP Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	El identificador único de grupo correspondiente al grupo que acabamos de crear.

4.6.8. Comando de creación de peticiones

El comando CMD_INSERT_PETITION se envía cuando se crea un nuevo grupo y se comienzan a añadir las peticiones de grupo al resto de alumnos que se han ido añadiendo. Para ello, se envía el identificador de grupo que se recibe del comando CMD_INSERT_GROUP y, además, se añade el nombre del nuevo grupo y el alumno. En la Tabla 20, se pueden ver los parámetros enviados por el cliente al servidor.

Tabla 20. Comando CMD_INSERT_PETITION Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno
groupID	INTEGER	El identificador único de grupo
groupName	CHAR	El nombre del grupo

En este caso, únicamente se debe enviar una confirmación de si el proceso se ha llevado a cabo o no. En la Tabla 21, se ven los parámetros enviados al cliente.

Tabla 21. Comando CMD_INSERT_ PETITION Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	CHAR	"OK" si se añadió correctamente a la tabla. "NOK" en caso contrario.

El proceso de creación de un grupo con sus respectivas peticiones, se muestra en la Ilustración 16.

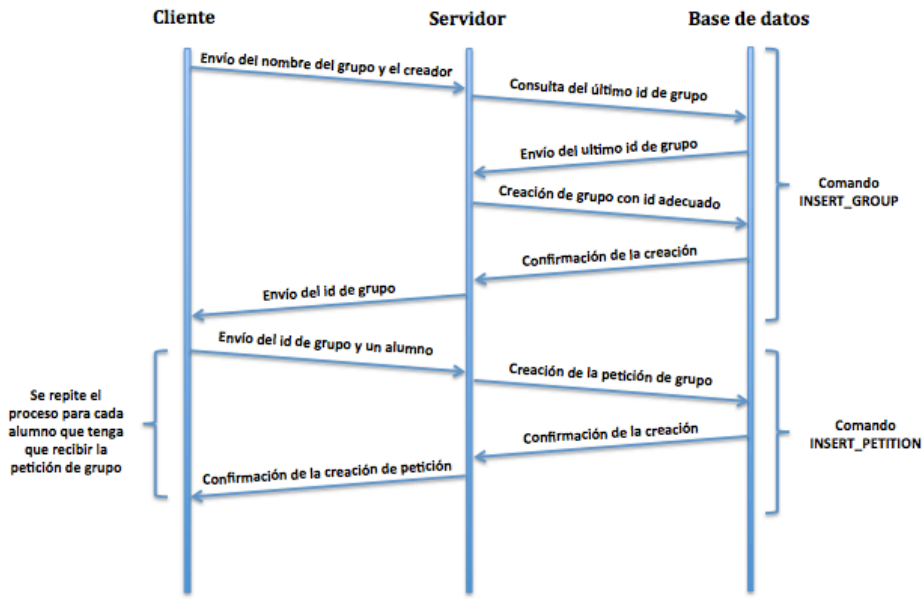

Ilustración 16. Proceso de creación de un grupo

4.6.9. Comando de obtención de nombre de grupo

El comando CMD_GET_GROUP_NAME se utiliza en la sección de 'Información', dentro del contenido específico de un grupo.

Como el control de los grupos se realiza a través de su identificador, cuando se quiere mostrar el nombre del grupo en 'Información' se necesita consultarlo a la base de datos. En la Tabla 22, se ve el formato del envío.

Tabla 22. Comando CMD_GET_GROUP_NAME Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo

Después de consultar la base de datos, el servidor envía al usuario el resultado de la búsqueda que incluye el nombre del grupo (Tabla 23).

Tabla 23. Comando CMD_ GET_GROUP_NAME Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	CHAR	Nombre del grupo solicitado

4.6.10. Comando de obtención de alumnos

El comando CMD_GET_ALUMNOS se solicita para poder cargar la lista de alumnos pertenecientes a un grupo.

Se envía el identificador de grupo y se busca en la base de datos a todos los alumnos asociados al grupo. El envío contiene los parámetros, recogidos en la Tabla 24.

Tabla 24. Comando CMD_GET_ALUMNOS Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo

El servidor no devuelve una respuesta única como ya sucede en otros casos, sino que envía una serie de respuestas, todas ellas unidas en un 'JsonArray' (sucesión de objetos 'Json' unidos en un 'Array'). Cada uno de estos paquetes 'Json' incluirá la información de un único alumno. El formato de estos paquetes se puede ver en la Tabla 25. Cada una de ellas corresponderá con la información de cada grupo del que el usuario ha recibido petición.

Tabla 25. Comando CMD_GET_ALUMNOS Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	CHAR	Nombre de un alumno

4.6.11. Comando para dejar un grupo

El comando CMD_LEFT_GROUP se envía para eliminar a un alumno de un grupo y borrar todas las referencias al mismo para ese grupo. En la Tabla 26, se puede ver lo necesario para esta petición, el identificador del grupo que queremos abandonar y el nombre del usuario que lo abandona.

Tabla 26. Comando CMD_LEFT_GROUP Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	NIA del alumno
groupID	INTEGER	El identificador único de grupo

El servidor se encarga de la eliminación de la información referente al alumno: chats pendientes del grupo destinados al usuario y el registro de grupo en la tabla 'GROUPS'. Además, en caso de que sea el último alumno del grupo, el servidor borrará todos los registros residuales de la base de datos de 'CHAT' referentes al grupo. También eliminará todas las entradas de la base de datos de 'AGENDA'. De esta manera, no quedará rastro alguno del grupo y se evita almacenar información inútil. Una vez realizadas todas estas comprobaciones y tareas, se envía una respuesta de confirmación al cliente, tal como se ve en la Tabla 27.

Tabla 27. Comando CMD_LEFT_GROUP Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si el proceso se realiza con éxito. "NOK" en caso contrario.

4.6.12. Comando para añadir una cita

El comando CMD_INSERT_AGENDA se envía cuando se intenta añadir una nueva cita.

Cada cita contiene ciertos parámetros que deben rellenarse obligatoriamente y que son enviados. Estos pueden verse en la Tabla 28:

Tabla 28. Comando CMD_INSERT_AGENDA Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo
lugar	CHAR	El lugar de la cita
asunto	CHAR	Motivo de la cita
fecha	CHAR	Cuándo será la cita
hora*	CHAR	Hora concreta a la que será la cita

En este comando existe una excepción que no se da en ningún otro. El último parámetro que el servidor espera recibir, "hora", no siempre se envía. No es un campo obligatorio para el paquete, ya que los terminales iOS no la envían. El servidor necesita el parámetro para los terminales Android y, por ese motivo, el servidor espera recibirlo. Por ello, existe una comprobación especial que extrae el parámetro del paquete 'Json'. En esta comprobación, si no se recibe se establece un valor por defecto. Una vez más, el servidor enviará una respuesta de confirmación como se ve en la Tabla 29.

Tabla 29. Comando CMD_ INSERT_AGENDA Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si la cita se añade con éxito. "NOK" en caso contrario.

4.6.13. Comando para obtener las citas

El comando CMD_GET_AGENDA se envía nada más entrar en la sección de Agenda para recibir el listado de citas. Para ello, se envía el identificador del grupo como se ve en la Tabla 30.

Tabla 30. Comando CMD_GET_AGENDA Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
groupID	INTEGER	El identificador único de grupo

El servidor no devuelve una respuesta única como sucede en otros casos, sino que envía una serie de respuestas, todas ellas unidas en un 'JsonArray' (sucesión de objetos 'Json' unidos en un 'Array'). Cada uno de estos paquetes 'Json' incluirá el nombre de una única cita. El nombre que tomamos como título de la cita y que por tanto enviamos, es el campo de "Asunto" de la base de datos. La información contenida en cada paquete se puede ver en la Tabla 31.

Tabla 31. Comando CMD_GET_AGENDA Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
asunto	CHAR	Título que se le da a la cita

4.6.14. Comando que obtiene detalles de una cita

El comando CMD_GET_AGENDA se envía, cuando se selecciona una cita de la lista de citas, para que se muestre la información de la misma. Se envía al servidor el identificador de grupo y el título de la cita como se muestra en la Tabla 32.

Tabla 32. Comando CMD_GET_AGENDA Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
asunto	CHAR	El parámetro que aparece como título de la cita
groupID	INTEGER	El identificador único de grupo

En este caso, ocurre algo parecido a cuando se inserta una cita en la base de datos (CMD_INSERT_AGENDA). Se envia toda la información almacenada de la cita, incluido el valor de 'hora', y será el cliente quien deberá considerar si tener en cuenta o no el valor recibido; ya que, fruto de la necesidad de estandarizar los envíos, se deben enviar todos los parámetros. Por todo ello, el paquete 'Json' resultante tendrá los parámetro mostrados en la Tabla 33.

Tabla 33. Comando CMD_GET_AGENDA Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
lugar	CHAR	El lugar de la cita
asunto	CHAR	Motivo de la cita
fecha	CHAR	Cuándo será la cita
hora	CHAR	Hora concreta a la que será la cita

4.6.15. Comando para recibir chats pendientes

El comando CMD_GET_CHAT se utiliza para solicitar el envío de todos los chats atrasados del alumno. Como se ve en la Tabla 34, el cliente envía el alumno y el grupo al que pertenece.

Tabla 34. Comando CMD_GET_CHAT Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	El alumno del que se quieren los chats
groupID	INTEGER	El identificador único de grupo

El servidor recopilará los chats pendientes del alumno enviado y los devolverá. Una vez más, no devuelve una respuesta, sino que envía una serie de respuestas, todas ellas unidas en un 'JsonArray' (sucesión de objetos 'Json' unidos en un 'Array'). Cada uno de estos paquetes 'Json' incluirá el nombre del que escribió el mensaje, para quién va dirigido(deberá coincidir con el alumno enviado por el cliente), la fecha en que se escribió el mensaje y el mensaje en cuestión. Todo ello con el formato mostrado en la Tabla 35.

Tabla 35. Comando CMD_GET_CHAT Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
user	CHAR	El alumno que escribió el mensaje
destino	CHAR	A quién va dirigido el mensaje
fecha	CHAR	Cuándo se escribió el mensaje
mensaje	CHAR	Contenido del mensaje

En el caso del Chat, existen ciertas diferencias entre el tratamiento que se da a las variables. En los dispositivos iOS, la "fecha" no se tiene en cuenta, ya que el principal motivo de su utilización es la ordenación, motivo que se compensa con la ordenación que de por sí tienen los paquetes 'Json' al enviarse en el 'JsonArray'.

4.6.16. Comando al enviar un chat

El comando CMD_INSERT_CHAT se utiliza cuando se quiere enviar algún mensaje. El funcionamiento habitual de este comando es ir acompañado del comando CMD_GET_CHAT. Para unificar código y evitar crear dos modos diferentes de inserción de un mensaje, se optó por generar una entrada en la tabla de chats pendientes también para el alumno que genera el mensaje; es decir, aparte de los registros de peticiones pendientes para el resto de compañeros, se generará una en la que tanto el que escribió como el destino sean el mismo alumno. De esta manera, una vez insertado el mensaje en la base de datos, se llama al comando para recuperar los mensajes pendientes desde el que llegará el mensaje recién introducido. En la Tabla 36, se pueden ver los parámetros necesarios para el envío de un mensaje.

Tabla 36. Comando CMD_INSERT_CHAT Envío de cliente

PARÁMETRO	TIPO	COMENTARIOS
username	CHAR	El alumno que escribió el mensaje
groupID	CHAR	El grupo en el que se envía el mensaje
destino	CHAR	A quién va dirigido el mensaje
fecha	CHAR	Cuándo se escribió el mensaje
mensaje	CHAR	Contenido del mensaje

En el caso de iOS, no se envía el parámetro "fecha", por lo que el servidor se ocupa de establecer el valor para cuando los terminales Android soliciten los mensajes. Como respuesta, según muestra la Tabla 37, sólo se confirma la inserción.

Tabla 37. Comando CMD_ INSERT_CHAT Respuesta del servidor

PARÁMETRO	TIPO	COMENTARIOS
result	INTEGER	"OK" si el mensaje se añade con éxito. "NOK" en caso contrario.

Como se puede ver, este comando deberá enviarse una vez por cada alumno del grupo, lo que provoca una unión de varios comandos para poder llevar a cabo la tarea:

- CMD_GET_ALUMNOS para obtener la lista de alumnos pertenecientes al grupo.
- CMD_INSERT_CHAT para enviar el mensaje a cada uno de ellos
- CMD_GET_CHAT para recargar los mensajes que se muestran.

4.7. Comunicación del servidor con la base de datos

Cada vez que el servidor precise alguna información que deba obtener de la base de datos, debe realizar una conexión a ella y realizar peticiones o 'Querys' (cuyo significado es 'Consultas').

Las consultas utilizan lenguaje SQL (Structured Query Language)[17], un lenguaje declarativo de acceso a Bases de Datos que permite especificar diferentes operaciones en ellas. Una de sus características principales es el uso de álgebra y cálculo relacional, lo que permite hacer consultas para recuperar o modificar información de manera sencilla en las Bases de Datos.

Dependiendo de la 'Query', se deberá especificar el contenido específico que queremos (a veces se solicita todo el contenido de un registro), la tabla en la que queremos consultar o el parámetro clave que delimitará una búsqueda. Existen

diferentes tipos de 'Querys' en función de la peticiones que queramos realizar. Las utilizadas en la aplicación son:

- SELECT: se trata estrictamente de una consulta. No modifica información ni añade ni elimina. Simplemente, devuelve los valores encontrados en la base de datos que concuerden con los parámetros indicados. Por ejemplo:
  `SELECT password FROM 'USERS' WHERE name="100066590";`
  Esta consulta devolvería el valor de la contraseña para el alumno "100066590".
  Este tipo de consulta se utiliza en la aplicación; por ejemplo, para el proceso de 'login', recuperaciones de listados o comprobaciones de usuarios.

- INSERT: se utiliza para añadir registros a una base de datos. En ella deben especificarse todos los campos que posea la tabla donde vayamos a insertar. Por ejemplo:
  `INSERT INTO USERS (id,name,pass) VALUES (1,'100066590','pass');`
  En la aplicación se utiliza por ejemplo al añadir grupos o peticiones de grupo.

- DELETE: esta sentencia eliminará registro de la base de datos que le especifiquemos. Por ejemplo:
  `DELETE FROM 'USERS' WHERE name="100066590";`
  Utilizadas en la aplicación para, por ejemplo, eliminar a un alumno de un grupo.

Las 'Querys' se definen en el servidor cuando se hace la conexión a la base de datos. El proceso simplificado de la conexión puede verse en la Ilustración 17.

```
//Database
Connection con = null;
Statement stmt = null;
ResultSet rs = null;
try{

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    con = DriverManager.getConnection("Ruta a la base de datos");
    stmt = con.createStatement();
    rs = stmt.executeQuery("query");


}catch (SQLException e){

}catch (ClassNotFoundException e){

} catch (InstantiationException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IllegalAccessException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Ilustración 17. Conexión del servidor con base de datos

La conexión con una base de datos concreta no se ve en la ilustración, sino que se muestra una simplificación genérica. Lo mismo ocurre con la declaración de la 'Query', que debería ir en la línea:

```
rs = stmt.executeQuery();
```

La declaración de las 'Querys' presenta una variación en función de la consulta que se quiera hacer:

- `stmt.executeQuery()`: si se trata estrictamente de una consulta como son las de tipo SELECT.
- `stmt.updateQuery()`: si se modifica la tabla de alguna manera (tipo INSERT o DELETE).

La razón es muy sencilla. En el primer caso, debemos almacenar los valores que devuelva la consulta. En el segundo, simplemente se recibe confirmación. Son respuestas distintas que no pueden ser tratadas de la misma manera. De hecho, en el segundo caso, la variable que recoge la respuesta es de tipo 'Integer', en vez de ser de tipo 'ResultSet'.

4.8. Servidor

La función principal del servidor es realizar la función de intermediario entre el cliente y la base de datos. Para esta finalidad, debe recibir peticiones del cliente y realizar consultas en la base de datos.

Para llevar a cabo esta función, en primer lugar, debe tener definidos los distintos tipos de peticiones o comandos que recibirá del cliente. Cómo se definen estos comando se puede ver en la Ilustración 18.

```
private static final String CMD_TEST = "test";
private static final String CMD_LOGIN = "login";
private static final String CMD_GET_GROUPS = "getGroups";
private static final String CMD_GET_PETITIONS = "getPetitions";
private static final String CMD_ACCEPT_PETITIONS = "acceptPetitions";
private static final String CMD_CHECK_PETITIONS = "checkPetitions";
private static final String CMD_CHECK_PARTNER = "checkPartner";
private static final String CMD_INSERT_GROUP = "insertGroup";
private static final String CMD_INSERT_PETITION = "insertPetition";
private static final String CMD_GET_GROUP_NAME = "getGroupName";
private static final String CMD_GET_ALUMNOS = "getAlumnos";
private static final String CMD_INSERT_AGENDA = "insertAgenda";
private static final String CMD_GET_AGENDA = "getAgenda";
private static final String CMD_GET_QUEDADA = "getQuedada";
private static final String CMD_LEFT_GROUP = "leftGroup";
private static final String CMD_GET_CHAT = "getChat";
private static final String CMD_INSERT_CHAT= "insertChat";
```

Ilustración 18. Definición de las peticiones que se reciben en el servidor

El comando CMD_TEST no se pide desde el cliente, es una comprobación que se puede realizar desde un navegador web y que informa de si el servidor se encuentra activo.

Como todas las peticiones recibidas desde el cliente transportan información para el servidor, todas ellas serán peticiones de tipo 'POST'. Por tanto, deberemos recibirlas en el método 'doPost()' del servidor. Un ejemplo de cómo se comprueba qué comando recibimos se muestra en la Ilustración 19.

```
public void doPost(HttpServletRequest request,HttpServletResponse response) throws
    ServletException,IOException{

    try {
        if(CMD_LOGIN.equalsIgnoreCase(request.getQueryString())){

            //Fin de CMD_LOGIN

        }else if(CMD_GET_GROUPS.equalsIgnoreCase(request.getQueryString())){

            //Fin de CMD_GET_GROUPS

        }else if(CMD_GET_PETITIONS.equalsIgnoreCase(request.getQueryString())){

            //Fin de CMD_GET_PETITIONS

        }else if(CMD_INSERT_PETITIONS.equalsIgnoreCase(request.getQueryString())){

            ...

}
```

Ilustración 19. Gestión de la recepción de comandos

Una vez establecido qué tipo de comando recibe, deberá tratar el paquete 'Json' que se recibe. Para ello, deberá solicitar recibir el paquete y luego almacenarlo en un objeto 'Json' para poder manipularlo. Un ejemplo de este proceso se puede ver en la Ilustración 20.

```
BufferedReader reader = request.getReader();
StringBuilder sb = new StringBuilder();
String line = reader.readLine();
while (line!=null){
    sb.append(line + "\n");
    line = reader.readLine();
}
reader.close();
String data = sb.toString();
JSONObject userjson = JSONObject.fromObject(data);
```

Ilustración 20. Tratamiento del paquete Json recibido en el servidor

Con el objeto 'Json' almacenado, se realizará la consulta la base de datos para obtener los datos requeridos. Dependiendo del comando en el que se esté, las consultas serán diferentes y, por tanto, los resultados obtenidos de esas consultas

también serán distintos. Por ello, dependiendo de la respuesta, el servidor formará dos tipos de respuesta distintas:

- Json: envía un paquete 'Json' con una respuesta (Ilustración 21).

```
//Respuesta al Json
PrintWriter toClient = response.getWriter();
JSONArray stringList= new JSONArray();
try {
    //Creamos un objeto de tipo JSON
    JSONObject cadena = new JSONObject();
    cadena.put("result",resultado);
    stringList.add(cadena1;
    toClient.println(stringList);
} catch (JSONException e) {
    e.printStackTrace();
}
```

Ilustración 21. Envío de un paquete Json simple desde el servidor

Aunque se envíe un 'Json' incluido en un 'JsonArray', para el receptor estará recibiendo un único objeto 'Json'.

- JsonArray: existen casos en los que hay que enviar más de un mismo resultado; es decir, enviar los mismos parámetros varias veces. Esto ocurre, por ejemplo, en peticiones como las de listas de grupos o de alumnos. En la Ilustración 22, se ve claramente el proceso.

```
//Obtenemos los resultados y los almacenamos
JSONArray stringList= new JSONArray();
while (rs.next()){//RS es la recepcion de BD
    JSONObject cadena = new JSONObject();
    cadena.put("groupId",rs.getObject(1).toString());
    cadena.put("groupName",rs.getObject(2).toString());
    stringList.add(cadena);
}
//Enviamos el JsonArray
PrintWriter toClient = response.getWriter();
try {
    toClient.println(stringList);
} catch (JSONException e) {
    e.printStackTrace();
}
```

Ilustración 22. Envío de un paquete JsonArray desde el servidor

Además de estas operaciones, en algunas peticiones realiza funciones extra. Puede realizar varias consultas, hacer comparaciones entre parámetros recibidos del cliente y datos obtenidos en la base de datos, etc.

Existe una petición que ya se ha comentado y que no se recibe del cliente. Se trata de la petición de control que puede realizarse desde cualquier navegador web. Como en este caso el servidor no recibe ningún tipo de información, simplemente es una consulta genérica, este tipo de consulta no puede tratarse igual que los comandos anteriores que se trataban en el método 'doPost()'. Este caso debe tratarse en un método aparte llamado 'doGet()', ya que esta petición es

del tipo 'GET'. En la Ilustración 23, se puede ver cómo se recibe el comando en su método adecuado:

```
public void doGet(HttpServletRequest request,HttpServletResponse response)
    throws ServletException,IOException{

    if(CMD_TEST.equalsIgnoreCase(request.getQueryString())){

        //Metodo test

    }
}
```

Ilustración 23. Recepción de la petición de tipo GET

Dentro del método se realiza el envío de un paquete 'Json' como ya se ha visto, con una respuesta estándar: "OK".

Aparte de la recepción de peticiones, era necesario, para evitar el almacenamiento de datos pasados, que el servidor borrase datos cada cierto tiempo. Dispone de un método que comprueba periódicamente la fecha actual, con las fechas introducidas en la tabla de 'AGENDA' de la base de datos. De esta manera, cuando encuentra una cita que ya se ha producido, la borra de la base de datos.

Para realizar el proceso de revisión, se ha creado una nueva clase cuyos objetos tienen un 'Timer' (objeto que permite realizar acciones cada un cierto tiempo dado) integrado al que se le define su método 'run()', donde se realizan las tareas que se requieran. En este caso, en el método 'run()', se incluirá la consulta a la 'AGENDA', la comparación con la fecha actual y, en caso de ser necesario, la eliminación de registros de la base de datos. Al inicio del servidor, se define un objeto de la nueva clase y el objeto 'Timer' incluido empezará a trabajar. El esquema sencillo y genérico de la nueva clase puede verse en la Ilustración 24.

```
public class DeleteOldEntries {

    //Declaracion del objeto
    public DeleteOldEntries(){
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask(){
            public void run() {

                //Funcion requerida

            }
        },0L,10000L);//Tiempo de refresco
    }
}
```

Ilustración 24. Clase que ejecuta acciones cada cierto tiempo

4.9. Conclusiones

La implementación de todo el sistema se estructuró en tres grandes bloques (en la sección 4.2. de este capítulo). El segundo de ellos, equivalente a la implementación del servidor y la base de datos, se realizó en estrecha colaboración con el alumno David Argüeso González (NIA:100066661). Este bloque forma parte tanto de su proyecto como de éste.

Fruto de la colaboración con David Argüeso González, el servidor se implementó de manera que fuera utilizable para dispositivos iOS y Android. Además, teniendo en cuenta esta necesidad de diseño, se aprovechó la oportunidad para realizar la implementación general del servidor utilizable para todo tipo de dispositivos, no sólo los incluidos en los proyectos.

Capítulo 5. Pruebas

5.1. Introducción

Una de las partes más importantes en el desarrollo de cualquier proyecto son las pruebas de funcionamiento. Estas pruebas sirven de comprobación de requisitos. Además de comprobar el correcto funcionamiento en función de las especificaciones iniciales, permiten descubrir deficiencias o casos que los requisitos iniciales no contemplaban.

En primer lugar, se establecen las condiciones específicas en las que se llevarán a cabo las pruebas de funcionamiento. En segundo lugar, se detalla el resultado de las pruebas según el esquema de objetivos detallados en el capítulo de diseño. Por último, se repasan los resultados de las pruebas y se comparan con los requisitos.

5.2. Condiciones de las pruebas

- Dispositivo de Pruebas:
  - iPhone 4 versión de iOS: 6.1.3.
- Servidor:
  - Máquina Virtual(Oracle VM VirtualBox).
    - Versión de VirtualBox: 4.2.8.
    - Oracle Java 7 (JDK).
    - Sistema Operativo: Ubuntu 12.04.
    - Memoria base: 1024MB.
    - Memoria de video: 64MB.
    - Tamaño Virtual: 10GB.
  - Apache tomcat 7.0.37

Las pruebas se realizaron con un único dispositivo de pruebas, aunque fueron diversos usuarios los que llevaron a cabo las pruebas.

Los usuarios que llevaron a cabo las pruebas recibieron un informe detallado de los requisitos de funcionamiento. Fueron asistidos por el desarrollador en todo momento durante las pruebas.

Las comprobaciones no se llevaron a cabo únicamente a través del funcionamiento de la aplicación, sino que se comprobó el tráfico a través del servidor, para comprobar el buen funcionamiento de éste, y la interactividad con la base de datos, comprobando el correcto estado de las modificaciones o consultas realizadas.

5.3. Pruebas de funcionamiento en la zona general

5.3.1. Login inicial de usuario

Si el usuario no introduce ningún valor en los campos o introduce valores que no se correspondan con los datos de usuario en la base de datos, debe mostrar mensajes de error, pero mensajes diferentes (Ilustración 25).

Ilustración 25. Pruebas de 'Login'

5.3.2. Pruebas de la pantalla de Listado de grupos

Los grupos a los que se pertenece se muestran en la pantalla principal. Si existen peticiones de grupo pendientes debe indicarse (Ilustración 26).

Ilustración 26. Pruebas de Listado de grupo

5.3.3. Pruebas de peticiones de grupo

Cuando un alumno crea un nuevo grupo, para los alumnos incluidos en el mismo, se crea la entrada pertinente en la tabla de 'Peticiones' y en la pantalla del listado de Peticiones de los alumnos aparecen los grupos que se muestran en la base de datos como "Pendientes".

5.3.4. Pruebas de creación de grupo

Comprobación de alertas referentes a errores en la introducción de alumnos o campos vacíos. Comprobación de que al crear un grupo se realizan las llamadas adecuadas en el servidor y se incluyen los datos en las tablas adecuadas (Ilustración 27).

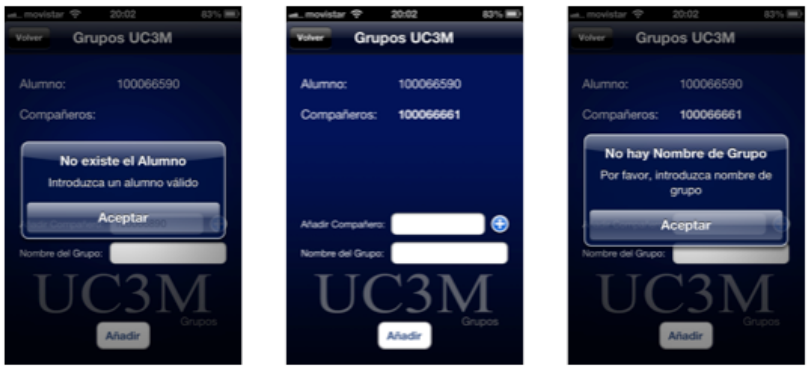

Ilustración 27. Pruebas de creación de grupo

5.4. Pruebas de funcionamiento en la zona específica

5.4.1. Pruebas de 'Chat'

Al acceder al 'Chat' se reciben los mensajes pendientes, se muestran y además se eliminan de la base de datos. Cuando enviamos un mensaje, se introducen los registros adecuados en la base de datos. El refresco de los mensajes actúa adecuadamente (Ilustración 28).

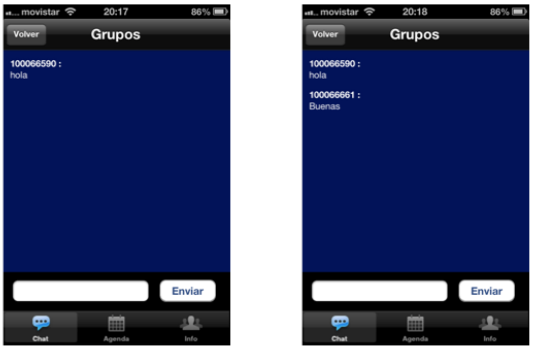

Ilustración 28. Pruebas de 'Chat'

5.4.2. Pruebas de 'Agenda'

Se muestra el listado de citas ya creadas existentes en la base de datos. No se permite dejar los campos sin rellenar al incluir una nueva cita.

La selección de la fecha de la cita se realiza con un "selector de fecha". Cuando un nueva cita es creada, se inserta correctamente en la base de datos. Las citas que sobrepasan la fecha actual se eliminan de la base de datos.

En la Ilustración 29, se puede ver un proceso de creación de cita, incluida la visualización de la misma una vez creada.

Ilustración 29. Pruebas de 'Agenda'

5.4.3. Pruebas de 'Información'

El nombre del grupo y los alumnos integrantes se muestran adecuadamente.

La opción de añadir a un nuevo integrantes cumple los requisitos de comprobación de alumno ya existente en el grupo y de ser un alumno válido.

Al abandonar el grupo, se elimina el archivo donde se guardan los mensajes del 'Chat' del grupo. Además, si es el último alumno, se eliminan de la base de datos todos los registros de la 'Agenda', todas las peticiones pendientes de grupo y todos los mensajes pendientes de ser enviados para el 'Chat'.

En la Ilustración 30, se muestran los mensajes recibidos al añadir un nuevo alumno al grupo y la alerta cuando se solicita salir del mismo.

Ilustración 30. Pruebas de 'Información'.

5.5. Cumplimiento de los requisitos

Resumen de los requisitos mínimos generales y su estado (Tabla 38):

Tabla 38. Revisión de los objetivos mínimos generales

	ESTADO
1. El proyecto debe abarcar la implementación de la aplicación, el servidor y la base de datos.	✓
2. La implementación debe ser compatible con móviles de distintos sistemas operativos	✓

Resumen de los requisitos mínimos de la aplicación y su estado (Tabla 39):

Tabla 39. Revisión de los objetivos de la aplicación

	A	B	C	D	E	F
1. Login	✓	✓	✓	-	-	-
2. Listado de grupos	✓	✓	✓	✓	-	-
3. Creación de grupo	✓	✓	✓	✓	✓	✓
4. Peticiones pendientes de grupo	✓	✓	-	-	-	-
5. Grupo	✓	✓	✓	✓	-	-
6. Chat	✓	✓	✓	✓	✓	✓
7. Agenda	✓	✓	✓	✓	✓	✓
8. Información de grupo	✓	✓	✓	✓	-	-

Todos los requisitos iniciales se cumplen. Además, se incluyen algunas mejoras de funcionamiento que en los requisitos iniciales no aparecían y que se detectaron durante las pruebas de funcionamiento:

- Mejoras de interfaz. Mucho más intuitiva ahora.
- Control de la respuesta de la aplicación cuando no existe conexión con el servidor.
- Borrado de citas con fecha antigua. No se guardan citas indefinidamente, sino que se comprueba si son citas pasadas y, de serlo, se borran.

Capítulo 6. Presupuesto

6.1. Planificación

Para el cálculo del tiempo invertido, se dividen las fases del proyecto en tablas. Finalmente, en la última tabla un resumen de la duración total.

En la Tabla 40, se muestran los tiempos de duración de la "Primera Parte".

Tabla 40. Planificación. Primera Parte

TAREA	DURACIÓN	COMIENZO	FIN
Diseño de la interfaz	5	Lun 01/10/12	Vie 05/10/12
Implementación de la interfaz simple	36	Lun 08/10/12	Lun 01/10/12
Pruebas de interfaz simple	4	Mar 27/11/12	Vie 30/11/12
TOTAL	**45**	**Lun 01/10/12**	**Vie 30/11/12**

En la Tabla 41, se muestran los tiempos de duración de la "Segunda Parte".

Tabla 41. Planificación. Segunda Parte

TAREA	DURACIÓN	COMIENZO	FIN
Diseño de la base de batos	10	Lun 04/02/13	Vie 15/02/13
Diseño del servidor	10	Lun 04/02/13	Vie 15/02/13
Implementación de base de datos	10	Lun 18/02/13	Vie 01/03/13
Implementación del servidor	10	Lun 18/02/13	Vie 01/03/13
Pruebas de la aplicación	10	Lun 04/03/13	Vie 15/03/13
TOTAL	**30**	**Lun 04/02/13**	**Vie 15/03/13**

En la Tabla 42, se muestran los tiempos de duración de la "Tercera Parte".

Tabla 42. Planificación. Tercera Parte

TAREA	DURACIÓN	COMIENZO	FIN
Arreglo de errores e introducción de mejoras sugeridas	10	Lun 18/03/13	Vie 29/03/13
Última ronda de pruebas	10	Lun 01/04/13	Vie 12/04/13
TOTAL	**20**	**Lun 18/03/13**	**Vie 12/04/13**

En la Tabla 43, se muestran los tiempos de duración de la "Cuarta Parte".

Tabla 43. Planificación. Cuarta Parte

TAREA	DURACIÓN	COMIENZO	FIN
Redacción de la Memoria	23	Mié 01/05/13	Vie 31/05/13
TOTAL	**23**	**Mié 01/05/13**	**Vie 31/05/13**

En la Tabla 44 se muestran los tiempos de duración totales.

Tabla 44. Planificación Total

TAREA	DURACIÓN	COMIENZO	FIN
Primera Parte	45	Lun 01/10/12	Vie 30/11/12
Segunda Parte	30	Lun 04/02/13	Vie 15/03/13
Tercera Parte	20	Lun 18/03/13	Vie 12/04/13
Cuarta Parte	23	Mié 01/05/13	Vie 31/05/13
TOTAL	**118**	**Lun 01/10/12**	**Vie 31/05/13**

Cabe destacar que, dentro de la planificación mostrada, se incluyen únicamente días laborables. Además, se realiza una estimación de horas trabajadas para la posterior redacción del 'Presupuesto'. No se considera que todos los días se dedicaron en exclusiva al Proyecto.

6.2. Diagrama de Gantt

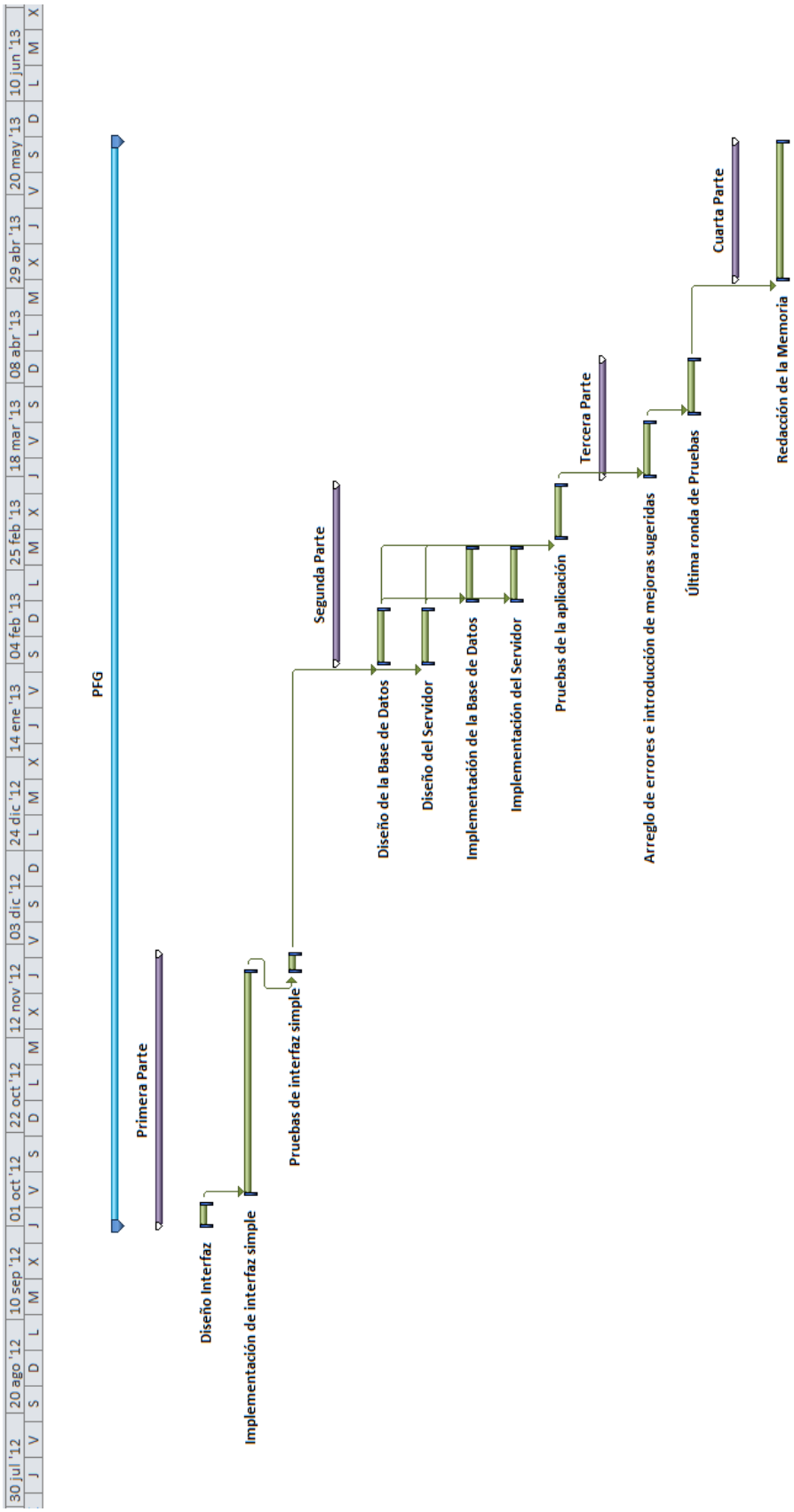

Ilustración 31. Diagrama de Gantt

6.3. Presupuesto

Una vez establecidos los tiempos de duración del proyecto en la 'Planificación', se muestra un Presupuesto detallado del coste necesario para llevarlo a cabo[21]. La Tabla 45 muestra los costes directos de personal.

Tabla 45. Desglose Presupuestario. Personal

APELLIDOS Y NOMBRE	CATEGORÍA	DEDICACIÓN (HOMBRE MES)*	COSTE HOMBRE MES (EUROS)	COSTE (EUROS)
Estévez Ayres, Iria Manuela	Ingeniero Sénior	0,5	4.289,54	2.144,77
Pozo Santiago, Juan Manuel	Ingeniero	3	2.694,39	8.083,17
Argüeso González, David	Ingeniero	0,7	2.694,39	1.886,07
			Total	**12.114,01**

*1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1.575 horas)

Dentro del personal, se incluye el trabajo realizado por el otro alumno que desarrolló la parte correspondiente a la aplicación en Android. El alumno participó en el desarrollo de la parte correspondiente al servidor y la base de datos. Por ello, se le incluye en el presupuesto con un tiempo de dedicación menor; ya que, el tiempo dedicado al desarrollo de la aplicación en Android no se tiene en cuenta.

La Tabla 46 muestra los costes directos de los equipos.

Tabla 46. Desglose Presupuestario. Equipos

DESCRIPCIÓN	COSTE (EUROS)	% USO DEDICADO PROYECTO	DEDICACIÓN (MESES)	PERIÓDO DE DEPRECIACIÓN	COSTE IMPUTABLE *
MacMini	1500,00	100	3	60	75,00
MacBook Pro	1200,00	100	3	60	60,00
iPhone 4	200	100	1	60	3,33
				Total	**138,33**

* Fórmula de cálculo de amortización: $\frac{A}{B} x C x D$

- A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
- B = periodo de depreciación (60 meses).
- C = coste del equipo (sin IVA).
- D = % del uso que se dedica al proyecto (habitualmente 100%).

A los gastos de personal y equipos hay que añadir los costes indirectos, que suponen un 20% de los costes directos. De esta manera, en la Tabla 47 se ve el coste total del proyecto.

Tabla 47. Presupuesto total

COSTES TOTALES	COSTES (EUROS)
Personal	12.114
Amortización	138
Costes indirectos	2.450
TOTAL	**14.703**

Capítulo 7. Conclusiones

7.1. Introducción

Con los requisitos cumplidos y con unas pruebas de funcionamiento satisfactorias, se puede dar por concluido el proyecto.

Es el momento de considerar el resultado obtenido en el proyecto, el margen de mejora del que todavía se dispone y las posibles implicaciones que puede conllevar en un futuro.

7.2. Resultado del Proyecto

La idea inicial y los objetivos propuestos se han superado con creces; no obstante, funcionalmente muchas partes del proyecto son mejorables.

De la misma manera que al realizar las pruebas se detectaron necesidades que no estaban incluidas en los requisitos iniciales, se detectaron también situaciones que, aunque no son necesarias para el correcto funcionamiento de la aplicación, facilitarían al usuario la comprensión de las acciones y harían más atractiva la interactividad.

Es posible que las opciones que ofrece la aplicación sean escasas; ya que, únicamente ofrece un servicio de mensajería y de citas, pero también hay que tener en cuenta que toda mejora e inclusión de servicios es posible dada la estructura actual.

La estructura general del Proyecto se planeó de manera que el resultado fuese fácilmente modificable. Por ello, sobre un sistema robusto como el que se ha implementado, establecer nuevos servicios no supondría modificar ninguna estructura existente, simplemente bastaría con añadir en el cliente, en el servidor y en la base de datos lo necesario.

Por las pruebas de funcionamiento, el cumplimiento de requisitos y por la viabilidad en la introducción de mejoras en el proyecto, el resultado general del proyecto se considera positivo.

7.3. Mejoras

El funcionamiento general de la aplicación cumple todos los requisitos propuestos inicialmente. Aun así, existen muchos aspectos que se deben mejorar, principalmente la gestión de errores de conexión con el servidor. Existen pequeños controles de conexión que comprueban si el usuario está recibiendo respuestas del servidor, pero en la mayoría de los casos, el usuario vería su navegación muy ralentizada o incluso paralizada en caso de que el servidor no se encuentre disponible.

Actualmente, la aplicación dispone de un tiempo de conexión, superado ese umbral, se considera que no puede conectar con el servidor y la conexión devuelve un mensaje de error. La aplicación controla ese error, pero el tiempo que permanece intentando conectar, ésta se bloquea. Además, debería realizarse un mejor control de estos errores de conexión.

Es necesario realizar más pruebas con distintos usuarios. Cuantos más usuarios y más variados, más errores se pueden descubrir y más mejoras descubrir. Es imprescindible, antes de plantearse cualquier salida al mercado o aprovechamiento por parte del público, comprobar el funcionamiento cuando el tráfico de usuarios es elevado. Las pruebas actuales han contemplado únicamente dos o tres usuarios conectados simultáneamente. No son pruebas realistas, ya que en cualquier entorno público se espera un volumen bastante mayor de conexiones simultáneas y, por tanto, es necesario saber como responderá el servidor ante esta circunstancia.

7.4. Futuro del Proyecto

El Servicio de Informática de la Universidad ha colaborado en la realización del Proyecto. Gracias a él, se han podido llevar a cabo pruebas de funcionamiento en dispositivos reales y no sólo a través de simulador; ya que, ha proporcionado los certificados y permisos necesarios exigidos por 'Apple Inc.'.

Esta colaboración está orientada a un trabajo conjunto y a una implementación del proyecto para la propia Universidad, permitiendo el uso de la aplicación a todos los alumnos de la Universidad Carlos III. Esto no será inmediato, ya que aunque la aplicación ha cumplido los requisitos y ha superado pruebas de funcionamiento, es necesario todavía realizar comprobaciones más complejas orientadas sobre todo al funcionamiento con un volumen alto de peticiones (múltiples usuarios).

Una vez superadas las pruebas, la aplicación desarrollada en este Proyecto pasará a formar parte de las distintas aplicaciones que ofrece el Servicio de Informática a los alumnos de la Universidad Carlos III.

Como fruto de esta colaboración, el Proyecto cobrará una dimensión mayor a la planificada inicialmente. Gracias a la independencia que proporciona la forma en que se ha implementado todo el sistema, se hará posible su exportación a otras universidades, necesitando únicamente unos pequeños retoques de diseño para adecuarlo a la Universidad de destino.

Capítulo 10. Bibliografía

1. Historia y actualidad de Apple Inc. (Fecha de consulta: 01-05-2013)
http://www.applegap.com/historia-de-apple/

2. Cuota de mercado de operadores
http://www.expansion.com/2013/04/17/empresas/digitech/1366223218.html

3. JSON (Fecha de consulta: 03-05-2013)
http://www.json.org/json-es.html

4. JSON Wikipedia (Fecha de consulta: 03-05-2013)
http://es.wikipedia.org/wiki/JSON

5. XML (Fecha de consulta: 03-05-2013)
http://www.slideshare.net/ocomur/xml-json-yaml

6. XML Wikipedia (Fecha de consulta: 03-05-2013)
http://es.wikipedia.org/wiki/Extensible_Markup_Language

7. Estructura de XML (Fecha de consulta: 11-05-2013)
http://www.cicei.com/ocon/gsi/tutorial_xml/Estructura.html

8. Servidores (Fecha de consulta: 11-05-2013)
http://es.wikipedia.org/wiki/Servidor

9. Servlets (Fecha de consulta: 11-05-2013)
http://www.edu4java.com/es/servlet/servlet1.html

10. Java Servlets (Fecha de consulta: 11-05-2013)
http://es.wikipedia.org/wiki/Java_Servlet

11. Interfaz de entrada común (Fecha de consulta: 11-05-2013)
http://es.wikipedia.org/wiki/Interfaz_de_entrada_com%C3%BAn

12. Bases de Datos Wikipedia (Fecha de consulta: 23-05-2013)
http://es.wikipedia.org/wiki/Base_de_datos

13. Bases de datos (Fecha de consulta: 23-05-2013)
http://www.maestrosdelweb.com/editorial/%C2%BFque-son-las-bases-de-datos/

14. ¿Qué es MySql? (Fecha de consulta: 24-05-2013)
http://indira-informatica.blogspot.com.es/2007/09/qu-es-mysql.html

15. MongoDB (Fecha de consulta: 24-05-2013)
http://es.wikipedia.org/wiki/MongoDB

16. MySQL (Fecha de consulta: 26-05-2013)
http://es.wikipedia.org/wiki/MySQL

17. Introducción a Querys (Fecha de consulta: 26-05-2013)
http://www.slideshare.net/alexysbike/introduccion-al-sql-query

18. NoSql (Fecha de consulta: 26-05-2013)
http://es.wikipedia.org/wiki/NoSQL

19. Consultas AdHoc (Fecha de consulta: 26-05-2013)
http://es.wikipedia.org/wiki/Ad_hoc

20. Servlets frente a CGI (Fecha de consulta: 30-05-2013)
http://boards5.melodysoft.com/M05/ventajas-de-los-servlets-sobre-el-79.html

21. Plantilla presupuesto UC3M (Fecha de consulta: 15-05-2013)
https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20%283%29_1.xlsx

22. *Ray J. Sams Teach Yourself iOS 5 Aplication Development in 24 Hours. USA: Pearson Education, Inc. 2012.*
ISBN-13: 978-0-672-33576-1
ISBN-10: 0-672-33576-X
(Fecha de consulta: 01-10-2012)

23. *Mark D., LaMarche J. Beginnig iPhone Development: Exploring the iPhone SDK. USA: Springer-Verlag New York, Inc.*
ISBN-13: 978-1-4302-1626-1
ISBN-10: 1-4302-1626-3
(Fecha de consulta: 01-10-2012)

24. Consultas de programación en sistema operativo iOS (1)
http://www.cocos2d-iphone.org
(Fecha de consulta: 01-10-2012)

25. Consultas de programación en sistema operativo iOS (2)
http://stackoverflow.com/questions
(Fecha de consulta: 01-10-2012)